université de BORDEAUX

LaBRI

Internship report

Towards an algebraic characterization of rational word functions

Author:
Nathan Lhote

Supervisors:
Emmanuel Filiot[1]
Olivier Gauwin[2]
Anca Muscholl[2]

June 23, 2015

[1]Université Libre de Bruxelles
[2]LaBRI, Université de Bordeaux

Introduction

In formal language theory, several models characterize regular languages: finite automata, congruences of finite index, monadic second-order logic (MSO), see *e.g.* [Büc60, Elg61, Tra61]. Further connections exist between monoid varieties and logical fragments of MSO, for instance aperiodic monoids have been shown to recognize exactly the word languages defined by first-order (FO) formulas [Sch65, MP71].

When considering word relations instead of languages, automata are replaced by transducers, that is automata with outputs associated with transitions. A transducer is an automaton that reads an input word and returns an output word obtained by concatenating the outputs of the transitions. However, several important properties do not generalize from automata to transducers. For instance, the well known equivalence between deterministic and non-deterministic one-way automata, as well as the equivalence between one-way and two-way automata, do not transfer to transducers. One property that is preserved is the equivalence between automata and MSO formulas: it has been shown [EH01] that MSO word transductions and two-way transducers define the same class of word functions called *regular* functions. A recently introduced model of computation, streaming string transducers (SST), has been shown to compute the same class of regular functions [AČ10]. Two recent related results are the equivalence between FO transductions and aperiodic SSTs [FKT14] and the equivalence between FO transductions and aperiodic two-way transducers [CD15]. However it is not known if one can decide if a given regular function is FO definable.

Our contribution: Our result deals with functions that are definable by one-way word transducers. These functions are known in the literature as *rational* functions.

The notion of minimal automaton goes beyond minimizing the state space. Indeed to decide whether a regular language satisfies some algebraic property, like aperiodicity, it suffices to consider the minimal automaton. Therefore in order to have an algebraic characterization of rational functions, we need a notion similar to the one of minimal automata for transductions. For the class of functions defined by deterministic transducers, such a notion exists [Cho03] and this minimal transducer

enjoys, among deterministic transducers, the same kind of minimality properties.

In an attempt to obtain a similar notion for rational functions, we study the model of bimachines [Sch61] which has been shown to be a canonical model for rational functions (see *e.g.* [BB79]). We describe a canonical bimachine, introduced by [RS91], and show that this representation preserves, similarly to the minimal automaton for languages, some algebraic properties of the function. In the case of aperiodic rational function, *i.e.* functions definable by an aperiodic one-way transducer, a characterization has already been given in [RS95]. Our main contribution is to give an effective characterization of aperiodic rational functions, and extend it to other algebraic varieties of rational functions. We also introduce *translations*, a model of logical transductions inspired by and equivalent to bimachines. We show an equivalence between transducers satisfying some algebraic property and logical translations, for instance between aperiodic transducers and FO-translations.

Contents

Chapter 1

Automata for languages and transductions

1.1 Words, languages and transductions

Let an *alphabet* Σ be a finite set of symbols called *letters*. Let Σ^* denote the free monoid on Σ. An element of Σ^* is called a *word* and the identity element ε is called the *empty word*. A *language* is a set of words.

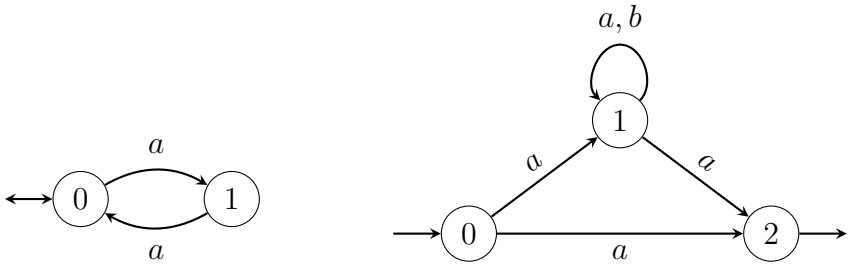
A *transduction* R is a subset of $\Sigma^* \times \Sigma^*$. For $(u, v) \in R$, u is called an *input* word and v an *output word*. The *domain* of R, denoted by $\mathrm{dom}(R)$, is the set of input words. A transduction is called *functional* if it is a (partial) function. For a functional transduction f we write $f(u) = v$ instead of $(u, v) \in f$.

Example 1.1.1. We give two running examples of functional transductions:

- Let $\Sigma = \{a\}$. The transduction $f_{even} : \Sigma^* \to \Sigma^*$ copies the word if its length is even and outputs ε otherwise.
- Let $\Sigma = \{a, b\}$. The transduction $f_{ends} : \Sigma^* \to \Sigma^*$ replaces each letter by a if the first and last letter are a and yields ε otherwise. *e.g.*, $f_{ends}(abaa) = a^4$, $f_{ends}(baaab) = f_{ends}(abab) = \varepsilon$.

1.2 Finite state automata

A *left non-deterministic finite state automaton* (NFA or simply automaton) over Σ is a tuple $A = (Q, I, F, \Delta)$, where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of accepting states, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation.



(a) DFA recognizing L_{even}. (b) NFA recognizing L_{ends}.

Figure 1.1: Two automata.

We denote $(p, \sigma, q) \in \Delta$ by $p \xrightarrow{\sigma}_A q$ (we will omit the A when the context is clear). We extend the relation to words: $p_0 \xrightarrow{\sigma_1 \dots \sigma_k} p_k$ when we have $p_0 \xrightarrow{\sigma_1} p_1 \xrightarrow{\sigma_2} p_2 \cdots p_{k-1} \xrightarrow{\sigma_k} p_k$. Such a sequence is called a *run of A on $\sigma_1 \dots \sigma_k$*, and if $p_0 \in I$ and $p_k \in F$, it is called a *successful run*. We also add $q \xrightarrow{\varepsilon} q$, $\forall q \in Q$.

A word $u \in \Sigma^*$ is *recognized* or *accepted* by A if there exist $p \in I$, $q \in F$ such that $p \xrightarrow{u} q$. The set of all words recognized by A is called the *language recognized* by A and is denoted by $L(A)$.

A *left deterministic finite state automaton* (left DFA or simply DFA) on an alphabet Σ is defined similarly to an NFA as a tuple $A = (Q, q_0, F, \delta)$ where the transition function is $\delta : Q \times \Sigma \to Q$ and $q_0 \in Q$ is the unique initial state.

Let $A = (Q, I, F, \Delta)$ be an automaton, and let $P_1, P_2 \subseteq Q$. Then we define $L_{P_1,P_2}(A)$ as the language recognized by $A' = (Q, P_1, P_2, \Delta)$.

Remark 1.2.1. Right automata are defined exactly as left automata except that they read the words from right to left. Let $A = (Q, I, F, \Delta)$ be a right automaton. Then we denote $(p, \sigma, q) \in \Delta$ by $q \xleftarrow{\sigma} p$ and $p_k \xleftarrow{\sigma_1 \dots \sigma_k} p_0$ when we have $p_k \xleftarrow{\sigma_1} p_{k-1} \cdots p_1 \xleftarrow{\sigma_k} p_0$. Unless specified otherwise, an automaton is assumed to be a left automaton.

Example 1.2.1. The DFA of Figure 1.1 (a) recognizes the language L_{even} of words of even length. The NFA of Figure 1.1 (b) recognizes the language L_{ends} of words over $\Sigma = \{a, b\}$ that start and end with an a.

1.3 Transducers

A *nondeterministic finite state transducer* (NFT) on an alphabet Σ is a tuple $T = (Q, I, F, \Delta, i, t)$, where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of accepting states, $\Delta : Q \times \Sigma \times Q \to \Sigma^*$ is a partial function

defining the transitions [1], $i : I \to \Sigma^*$ is the initial function. and $t : F \to \Sigma^*$ is the terminal function.

Let $\Delta(p, \sigma, q) = v$ be a transition of T. The letter σ and the word v are respectively called the *input* and the *output* of the transition. We denote $\Delta(p, \sigma, q) = v$ by $p \xrightarrow{\sigma|v}_T q$ (we will omit the T when the context is clear). We extend the relation to words: $p_0 \xrightarrow{\sigma_1 \dots \sigma_k | v_1 \dots v_k} p_k$ when we have $p_0 \xrightarrow{\sigma_1|v_1} p_1 \xrightarrow{\sigma_2|v_2} p_2 \cdots p_{k-1} \xrightarrow{\sigma_k|v_k} p_k$. Such a sequence is called a *run of* T *on* $\sigma_1 \dots \sigma_k$, and if $p_0 \in I$ and $p_k \in F$, it is called a *successful run*. We also add $q \xrightarrow{\varepsilon|\varepsilon} q$, $\forall q \in Q$. An NFT realizes a transduction noted $[\![T]\!] \subseteq \Sigma^* \times \Sigma^*$. We also say that T *defines* the transduction $[\![T]\!]$. The pair $(u, v) \in [\![T]\!]$ if there exist $q_0 \xrightarrow{u|w} q_f$ such that $q_0 \in I$, $q_f \in F$ and $v = i(q_o)\ w\ t(q_f)$. An NFT is called *functional* if it defines a functional transduction. From now on the NFTs considered will be functional.

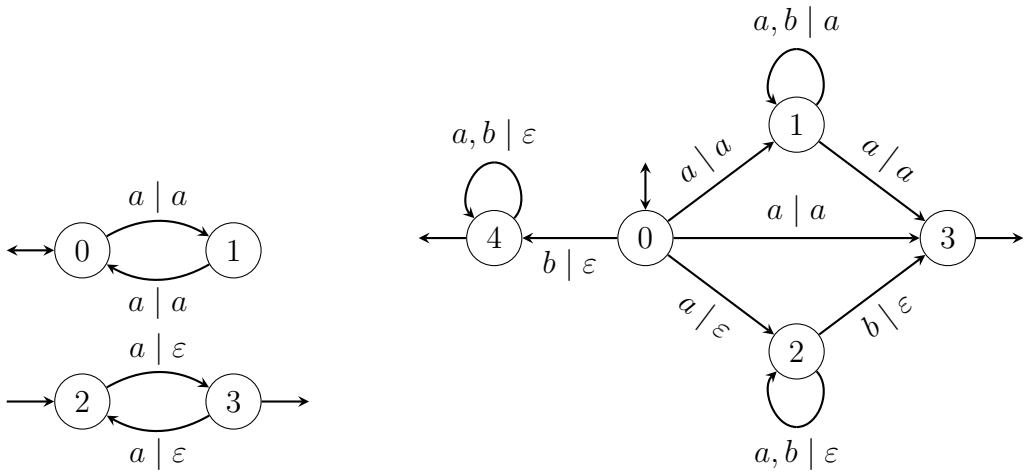
A *deterministic finite state transducer* (DFT) on an alphabet Σ is defined similarly to an NFT as a tuple $T = (Q, q_0, F, \delta, i, t)$ where the transition function is $\delta : Q \times \Sigma \to Q \times \Sigma^*$ and $q_0 \in Q$.

The *underlying automaton* of a transducer is the automaton obtained by removing the outputs from the transitions. Let $T = (Q, I, F, \Delta, i, t)$ be a transducer, and let $\bar{\Delta} \subseteq Q \times \Sigma \times Q$ be the projection of Δ. Then $A_T = (Q, I, F, \bar{\Delta})$ is called the underlying automaton of T.

Functions defined by NFTs are called *rational*, functions defined by DFTs are called *subsequential* and functions defined by DFTs without terminal output are called *sequential*.

Example 1.3.1. We give two examples of transducers: The transducer of Figure 1.2 (a) defines the function f_{even} while the transducer of Figure 1.2 (b) defines the function f_{ends}.

[1] This type of transducer is sometimes called *real-time*[Sak09]. In the general case, a transition of a transducer may be labelled by any pair of words



(a) NFT defining f_{even}.

(b) NFT defining f_{ends}.

Figure 1.2: Two transducers.

Chapter 2

Logical and algebraic characterizations of classes of regular languages

2.1 Congruences and monoids

Let Σ be a finite alphabet and let $\sim$ be an equivalence relation on Σ^*. We say that $\sim$ is a *right congruence* if it verifies:

$$\forall u, v \in \Sigma^*, \forall \sigma \in \Sigma, u \sim v \Rightarrow u\sigma \sim v\sigma$$

Symmetrically we define a *left congruence*. A *congruence* is defined as both a left and a right congruence.

Let $\sim_1$ and $\sim_2$ be two equivalence relations. We say that $\sim_1$ is *finer* than $\sim_2$, or that $\sim_2$ is *coarser* than $\sim_1$, denoted by $\sim_1 \sqsubseteq \sim_2$ if for all u, v we have $u \sim_1 v \Rightarrow u \sim_2 v$.

We recall that a monoid M is a set given with an associative binary operation (written multiplicatively) and an *identity element* which is neutral (both left and right) for this operation. A *monoid homomorphism* (morphism for short) is an application $\mu : M_1 \to M_2$ between two monoids verifying $\mu(xy) = \mu(x)\mu(y)$ for $x, y \in M_1$ and $\mu(1_{M_1}) = 1_{M_2}$ with 1_{M_i} being the identity element of M_i for $i \in \{1, 2\}$. We say that a language L of Σ^* is *recognized* by M if there exist $P \subseteq M$ and $\mu : \Sigma^* \to M$ a monoid morphism such that $L = \mu^{-1}(P)$. Let $\sim$ be a congruence on Σ^*. We say that L is *recognized* by $\sim$ if there exists $P \subseteq \Sigma^*/\sim$ such that $L = \{u \mid [u] \in P\}$.

Let M be a monoid and let $\mu : \Sigma^* \to M$. Then $\sim_\mu$ defined by $u \sim_\mu v \Leftrightarrow \mu(u) = \mu(v)$ is a congruence, and any language recognized by M is recognized by $\sim_\mu$ for some μ. Conversely, for a congruence $\sim$ we have a monoid $\Sigma^*/\sim$ with a binary operation induced by the natural morphism $u \mapsto [u]$ such that any language recognized by $\sim$ is recognized by $\Sigma^*/\sim$. Hence there is a natural equivalence between recognizability by congruence and by monoid.

Example 2.1.1. Let $\Sigma = \{a\}$. Let $\mu : \begin{array}{ccc} \Sigma^* & \to & \{0,1\} \\ a^n & \mapsto & n \bmod 2 \end{array}$ be the morphism from Σ^* into the group of two elements (written additively). Then $L_{even} = \mu^{-1}(0)$. The language L_{even} is recognized by the group of two elements.

The corresponding congruence is defined by $a^m \sim a^n$ iff $m = n \bmod 2$. Then $L_{even} = [\varepsilon]$. The language L_{even} is recognized by the congruence $\sim$.

2.1.1 Syntactic congruences and monoids

Let L be a language. We define the *syntactic congruence* of L as:

$$u \approx_L v \Leftrightarrow (\forall x, y \quad xuy \in L \Leftrightarrow xvy \in L)$$

From this we can naturally define the *syntactic monoid* of L, $M(L) = \Sigma^*/\approx_L$. According to the Myhill-Nerode Theorem [Ner58], a language is regular if and only if its syntactic congruence has finite index.

Remark 2.1.1. Let us also show that $\approx_L$ is the coarsest congruence recognizing L.

Proof: We see that, by definition, L is a union of classes modulo $\approx_L$ hence the syntactic congruence of L does recognize L. Let $\sim$ be a congruence recognizing L. Since $\sim$ is a congruence, if $u \sim v$ then for all $x, y \in \Sigma^*$, $xuy \sim xvy$. This means that if $u \sim v$, for all $x, y \in \Sigma^*$, $xuy \in L \Leftrightarrow xvy \in L$. Hence $u \approx_L v$ and $\sim$ is finer than $\approx_L$. □

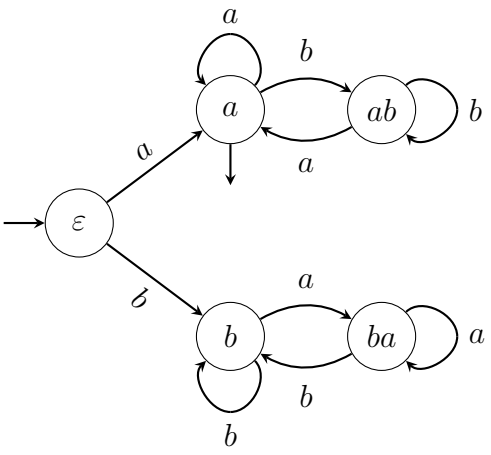
Example 2.1.2. Figure 2.1 (a) represents the multiplication table of the syntactic monoid of L_{ends}, with elements labelled as representatives of congruence classes (omitting the identity element).

Figure 2.1 (b) shows the automaton associated with the syntactic congruence of L_{ends}, with the states labelled as representatives of congruence classes.

The automaton associated with the syntactic congruence of L_{even} is simply the automaton of Figure 1.1. The syntactic monoid of L_{even} is the group with two elements.

	a	b	ab	ba
a	a	ab	ab	a
b	ba	b	b	ba
ab	a	ab	ab	a
ba	ba	b	b	ba

(a) Multiplication table of the syntactic monoid of L_{ends}.



(b) Automaton associated with the syntactic congruence of L_{ends}.

Figure 2.1

2.1.2 Transition congruences and monoids

Let A be an automaton, then we define the *transition congruence* of A as:

$$u \equiv_A v \Leftrightarrow (\forall p, q \in Q \quad u \in L_{p,q} \Leftrightarrow v \in L_{p,q})$$

From this we can naturally define the *transition monoid* of A, $M(A) = \Sigma^* / \equiv_A$.

Remark 2.1.2. Let A be an automaton. Then $L(A)$ is recognizable by $\equiv_A$. Indeed, if $u \equiv_A v$ and $u \in L(A)$ then $v \in L(A)$. Therefore $L(A)$ is the union of congruence classes. This also implies that $L(A)$ is recognizable by the monoid $M(A)$.

Let A be a left (resp. right) deterministic automaton, then we define the *right* (rep. *left*) *congruence associated with* A as:

$$u \sim_A v \Leftrightarrow (\forall p \in Q \quad u \in L_{q_0,p} \Leftrightarrow v \in L_{q_0,p})$$

Conversely, let $\sim$ be a right (resp. left) congruence. We can define a unique deterministic left (resp. right) automaton without final state associated with $\sim$ as: $A_\sim = (\Sigma^* / \sim, [\varepsilon], \delta)$ with δ being the natural right (resp. left) action of Σ on $\Sigma^* / \sim$.

Remark 2.1.3. Let A be a left (resp. right) automaton. If we assume that A is trim, *i.e.* any state appears in at least one successful run of A, then $u \equiv_A v$ if and only if for all $w \in \Sigma^*$, $wu \sim_A wv$ (resp. $uw \sim_A vw$).

Let us also remark that in that case, $\equiv_A$ is the coarsest congruence to be finer than $\sim_A$.

For a transducer T, we will use the notations $M(T)$, and $\equiv_T$ when referring to the transition monoid of the underlying automaton of T.

2.2 Monoid varieties

Let U be a finite set of variables and $s_1, s_2 \in U^*$. We say that a monoid M satisfies the equality $s_1 = s_2$ if for any morphism $\eta : U^* \to M$, we have $\eta(s_1) = \eta(s_2)$. We say that a congruence $\sim$ satisfies the equality $s_1 = s_2$ if for any morphism $\eta : U^* \to \Sigma^*$, we have $\eta(s_1) \sim \eta(s_2)$.

A *monoid pseudovariety* (variety for short) is defined [Str96] as a set of finite monoids that is closed under finite direct product, the taking of submonoids (*i.e.* a subset stable by multiplication containing the same identity element) and of homomorphic images. We will use the Theorem given in [ES76] to characterize monoid varieties. Let E be a countable set of equalities. Then we say that the set of finite monoids satisfying all but finitely many equalities in E is the monoid variety *ultimately defined* by E. The Theorem states that any monoid variety is ultimately definable. In the following we will use this characterization as a definition.

Let $\mathbf{V}$ be a monoid variety, we say that an automaton (resp. a transducer) is a ***V**-automaton* (resp. ***V**-transducer*) if its transition monoid is in $\mathbf{V}$. A language is called a ***V**-language* if there exists a $\mathbf{V}$-automaton recognizing it. A congruence is called a ***V**-congruence* if the monoid associated with it is in $\mathbf{V}$.

Proposition 2.2.1. *Let $\boldsymbol{V}$ be a monoid variety.*

- *Let $\sim$ be a congruence. Then $\sim$ is a $\boldsymbol{V}$-congruence if and only if it satisfies all but finitely many equalities defining $\boldsymbol{V}$.*
- *Let $\sim_1$, $\sim_2$ be two congruences such that $\sim_1 \sqsubseteq \sim_2$. If $\sim_1$ is a $\boldsymbol{V}$-congruence, then so is $\sim_2$.*

Proof: Let U be a finite set of variables. Let $M = \Sigma^*/\sim$ and let $\mu : \Sigma^* \to M$ be the natural morphism. Let us assume that $\sim$ is a $\mathbf{V}$-congruence. Let $\eta : U^* \to \Sigma^*$ be a morphism. Since $M \in \mathbf{V}$, we have by definition that if M satisfies $s_1 = s_2$ then $\mu \circ \eta(s_1) = \mu \circ \eta(s_2)$ which means that $\eta(s_1) \sim \eta(s_2)$. Conversely, let us assume that $\sim$ satisfies all but finitely many equalities of $\mathbf{V}$. Let $\eta : U^* \to M$ be a morphism. Let us show there exists a morphism $\nu : U^* \to \Sigma^*$ such that $\eta = \mu \circ \nu$. We only have to give the image of any $u \in U$ to describe a morphism. We define $\nu(u) = w_u$ with $w_u \in \mu^{-1}(\eta(u))$ (which is not empty since μ is surjective). This yields a morphism which verifies $\eta = \mu \circ \nu$. Hence we have that if $\sim$ satisfies $s_1 = s_2$ then $\nu(s_1) \sim \nu(s_2)$ which means that $\eta(s_1) = \eta(s_2)$.

Let $s_1, s_2 \in U^*$ such that $\sim_1$ satisfies $s_1 = s_2$, *i.e.* for any morphism $\eta : U^* \to \Sigma^*$, $\eta(s_1) \sim_1 \eta(s_2)$. In particular, $\eta(s_1) \sim_2 \eta(s_2)$, which means that $\sim_2$ satisfies $s_1 = s_2$. □

Remark 2.2.1. Some of the most commonly used varieties are shown in Figure 6.1. A particular case is the variety of aperiodic monoids, denoted by **A**, which has been shown to recognize the languages definable in first-order logic [Sch65, MP71].

For a given variety **V**, the *membership problem*, *i.e.* deciding if some monoid belongs to **V**, is not decidable *a priori*. For a discussion on this problem see [Pin97]. However, if the membership problem is decidable, then the problem of deciding if a language is a **V**-language is also decidable. Indeed, according to Remark 2.1.1 and Proposition 2.2.1, one only has to check that the syntactic monoid of the language belongs to **V**.

2.3 Logics

Given an alphabet Σ, *monadic second-order formulas* (MSO formulas) are built over first order variables $x, y, \ldots$ and second order variables $X, Y, \ldots$. Atomic formulas are built using the binary predicate $<$ and for each $\sigma \in \Sigma$ a unary predicate, also denoted by σ. The formulas are defined by the following grammar:

$$\varphi ::= \exists X\ \varphi \mid \exists x\ \varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid x \in X \mid \sigma(x) \mid x < y \mid (\varphi)$$

Universal quantifiers and other boolean connectives are defined as usual: $\forall X\ \varphi \equiv \neg\exists X\ \neg\varphi$, $\forall x\ \varphi \equiv \neg\exists x\ \neg\varphi$, $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$. We also define the constant formulas $\top$ and $\bot$ which are respectively always and never satisfied.

We do not define the semantics of MSO formulas, neither the standard notion of free and bound variables, but rather give examples and refer the reader to [EF95] for formal definitions.

Given a closed formula φ, and a word w satisfying φ, we write $w \models \varphi$ and $L(\varphi)$ denotes the language of words satisfying φ.

A *logical fragment* of MSO (logic for short) $\mathcal{F}$ is a subset of formulas of MSO. Given a language L, we say that L is $\mathcal{F}$-definable (or an $\mathcal{F}$-language) is there exists a closed formula $\varphi \in \mathcal{F}$ such that $L = L(\varphi)$.

Example 2.3.1. We define the successor relation as:

$$\mathsf{succ}(x, y) = (x < y) \wedge (\neg\exists z\ (x < z) \wedge (z < y))$$

We define the minimum and maximum unary predicates:

$$\mathsf{min}(x) = \neg\exists y\ (y < x)$$

$$\mathsf{max}(x) = \neg\exists y\ (x < y)$$

Then we define the MSO formula φ_{even} which defines the language of words of even length:

$$\begin{aligned}\varphi_{even} = \exists X\ \exists Y\ \forall x\ \forall y \quad & x \in X \leftrightarrow \neg(x \in Y) \\ \wedge\ & \mathsf{min}(x) \rightarrow x \in X \\ \wedge\ & \mathsf{max}(y) \rightarrow y \in Y \\ \wedge\ & \mathsf{succ}(x,y) \rightarrow (x \in X \wedge y \in Y) \vee (x \in Y \wedge y \in X)\end{aligned}$$

We give below several logics and refer to Figure 6.1 to see the languages defined by these logics.

First-order formulas (FO) denotes the logic of formulas without any second order variables.

Example 2.3.2. Let $\Sigma = \{a, b\}$. We define the formula φ_{ends} which defines the language L_{ends}:

$$\varphi_{ends} = \exists x\ \exists y\ \mathsf{min}(x) \wedge a(x) \wedge \mathsf{max}(y) \wedge a(y)$$

$MSO[=]$ denotes the logic of formulas without the binary predicate $<$.

Let k be a non-negative integer, then FO^k denotes the logic of FO formulas with only k variables.

Let $\mathbf{V}$ be a monoid variety and let $\mathcal{F}$ be a logic. We say that $\mathcal{F}$ is a logic *corresponding to* $\mathbf{V}$ if for any language L we have: L is an $\mathcal{F}$-language if and only if L is a $\mathbf{V}$-language.

Chapter 3

Logic-bimachine correspondence

3.1 Bimachines

Here we will describe the notion of a bimachine, introduced by Schützenberger in [Sch61], which is in some sense both left and right sequential (or subsequential). Bimachines have been shown, see *e.g.* [BB79], to define exactly rational functions and are in that sense equivalent to functional NFTs.

A bimachine is given by a tuple $B = (L, R, \omega, \lambda, \rho)$ where:

- L is the set of states of a deterministic left automaton given with an initial state l_0.
- R is the set of states of a deterministic right automaton given with an initial state r_0.
- The output function $\omega : L \times \Sigma \times R \to \Sigma^*$ which may be partial.
- The terminal left function $\rho : L \to \Sigma^*$ and the terminal right function $\lambda : R \to \Sigma^*$ which may both be partial.

The output function is extended to words in this fashion: Let $u = \sigma_1 \dots \sigma_n \in \Sigma^n$, let $l \xrightarrow{\sigma_1} l_1 \cdots l_{n-1} \xrightarrow{\sigma_n} l_n$ be an execution of L on u from l and let $r_n \xleftarrow{\sigma_1} r_{n-1} \cdots r_1 \xleftarrow{\sigma_n} r$ be an execution of R on u from r. Then, if it is defined:

$$\omega(l, u, r) = \omega(l, \sigma_1, r_{n-1}) \cdots \omega(l_{n-1}, \sigma_n, r)$$

If $l = l_0$ and $r = r_0$, when it exists the image of u by B is defined as:

$$[\![B]\!](u) = \lambda(r_n)\omega(l_0, u, r_0)\rho(l_n)$$

.

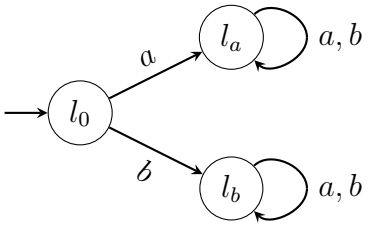


Figure 3.1: Left automaton of B.

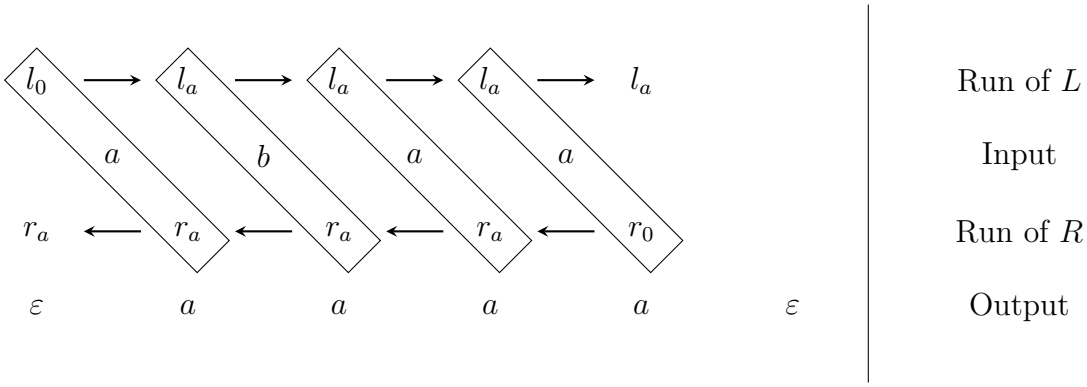


Figure 3.2: Execution of a bimachine

Remark 3.1.1. In order to simplify, we will write L (resp. R) when referring both to the set of states and the automaton.

Remark 3.1.2. When R is reduced to a single element then the bimachine is simply a subsequential transducer (sequential if ρ is a constant function equal to ε on its domain).

Example 3.1.1. We describe a bimachine $B = (L, R, \omega, \lambda, \rho)$ defining f_{ends}. The automata $L = \{l_0, l_a, l_b\}$ and $R = \{r_0, r_a, r_b\}$ are identical (except that one is a left automaton and the other is a right automaton) and L is shown in Figure 3.1. The functions λ and ρ are constant and equal to ε, $\omega(l, \sigma, r) = a$ if $l \in \{l_0, l_a\}$, $\sigma = a$ and $r \in \{r_0, r_a\}$ or if $l = l_a$, $\sigma = b$ and $r = r_a$. In the other cases $\omega(l, \sigma, r) = \varepsilon$. In Figure 3.2 we show the execution of this bimachine on the word *abaa*.

A bimachine is called *complete* if its output function is total.

Proposition 3.1.1. *Any bimachine is equivalent to some complete bimachine.*

Proof: The idea of the proof is to extend the output function by outputting ε

when it is not defined. To ensure that the domain of the function is preserved, we refine the left (for instance) automaton to only accept words in the domain.

Let $B = (L, R, \omega, \lambda, \rho)$ be a bimachine defining a function f. We construct $B' = (L', R, \omega', \lambda, \rho')$ a complete bimachine equivalent to B. Let A be an automaton recognizing $\mathrm{dom}(f)$, which is rational. We define $L' = L \times A$. Let (l, q) be a state of L' and let $r \in R$. Then $\omega'((l, q), \sigma, r) = \omega(l, \sigma, r)$ if it is defined and $\omega'((l, q), \sigma, r) = \varepsilon$ otherwise. If q is a final state of A then $\rho'(l, q) = \rho(l)$, otherwise $\rho'(l, q)$ is not defined.

If a word belongs to $\mathrm{dom}(f)$ then the outputs in the run of B' are the same as the ones in B. If a word does not belong to $\mathrm{dom}(f)$ then it is not accepted by L'. Hence B and B' are equivalent. □

3.1.1 From bimachines to transducers

Let $B = (L, R, \omega, \lambda, \rho)$ be a bimachine defining f, we can construct an NFT $T = (Q, I, F, \Delta, i, t)$ defining the same function:

- $Q = L \times R$
- $I = \{l_0\} \times \mathrm{dom}(\lambda)$.
- $F = \mathrm{dom}(\rho) \times \{r_0\}$.
- $\Delta((l_1, r_1), \sigma, (l_2, r_2)) = v$
  if $l_1 \xrightarrow{\sigma} l_2$, $r_1 \xleftarrow{\sigma} r_2$ and $\omega(l_1, \sigma, r_2) = v$.
- $i : (l_0, r) \mapsto \lambda(r)$ whenever $(l_0, r) \in I$.
- $t : (l, r_0) \mapsto \rho(l)$ whenever $(l, r_0) \in F$.

3.1.2 From transducers to bimachines

An NFA $A = (Q, I, F, \Delta)$ is called *unambiguous* is for any word $u = \sigma_1 \dots \sigma_n \in \Sigma^n$ there exists at most one sequence $q_0 \xrightarrow{\sigma_1} q_1 \cdots q_{n-1} \xrightarrow{\sigma_n} q_n$ with $q_0 \in I$ and $q_n \in F$. An NFT is called *unambiguous* if its underlying automaton is unambiguous. It has been shown that any functional NFT is equivalent to some unambiguous NFT [BB79].

Let $T = (Q, I, F, \Delta, i, t)$ be an unambiguous NFT, we can naturally describe an equivalent bimachine $B = (L, R, \omega, \lambda, \rho)$ with:

- $L = \Sigma^* / \equiv_T$ with the right action of Σ as its transition function: $[u] \xrightarrow{\sigma} [u\sigma]$ and $l_0 = [\varepsilon]$.

- $R = \Sigma^*/\equiv_T$ with the left action of Σ as its transition function: $[\sigma u] \xleftarrow{\sigma} [u]$ and $r_0 = [\varepsilon]$.
- $\omega(l, \sigma, r)$ is defined as the unique (by unambiguity) v such that $\exists u_l \in l, u_r \in r$ and $p, q \in Q$ with $u_l \in L_{I,p}$, $u_r \in L_{q,F}$ and $\Delta(p, \sigma, q) = v$.
- $\Delta((l_1, r_1), \sigma, (l_2, r_2)) = v$ if $l_1 \xrightarrow{\sigma} l_2$, $r_1 \xleftarrow{\sigma} r_2$ and $\omega(l_1, \sigma, r_2) = v$.
- $\lambda : \begin{array}{c} R \to \Sigma^* \\ [u] \mapsto i(p) \end{array}$ where p is the unique (by unambiguity) $p \in I$ such that $u \in L_{p,F}$.
- $\rho : \begin{array}{c} R \to \Sigma^* \\ [u] \mapsto t(p) \end{array}$ where p is the unique (by unambiguity) $p \in F$ such that $u \in L_{I,p}$.

3.2 Logical transduction

In the paper [Fil15] an equivalence is given between NFTs and so called *order-preserving* logical transducers. The following definition coincides with this class of order-preserving logical transductions, in the logics of MSO[<] and FO[<]. It is also somewhat similar to the notion of logical translation given by [MSTV06] in the particular case of length-preserving transductions.

Let $\mathcal{F}$ be a logic. An $\mathcal{F}$-translation is given by a tuple

$$\mathcal{T} = \left(k, S, \left(\varphi^{<}_{j,\sigma,v}, \varphi^{>}_{j,\sigma,v}\right)_{\substack{1 \le j \le k \\ \sigma \in \Sigma, v \in S}}, \left(\varphi^{i}_{v}\right)_{v \in S}, \left(\varphi^{t}_{v}\right)_{v \in S} \right)$$

$\mathcal{T}$ is composed of $k > 0$ and $S \subset \Sigma^*$ finite, the set of outputs. For $1 \le j \le k$, $\sigma \in \Sigma$ and $v \in S$, $\varphi^{<}_{j,\sigma,v}$, $\varphi^{>}_{j,\sigma,v}$, φ^{i}_{v} and φ^{t}_{v} are closed $\mathcal{F}$-formulas.

By definition $(u, w) \in [\![\mathcal{T}]\!]$ if:

> for $u = \sigma_1 \dots \sigma_n \in \Sigma^n$ there exists a decomposition $w = w_0 w_1 \dots w_n w_{n+1} \in S^{n+2}$ such that:
>
> - for $i \in \{1, \dots, n\}$ there exists j, with $1 \le j \le k$ such that we have both: $\sigma_1 \dots \sigma_{i-1} \models \varphi^{<}_{j,\sigma_i,w_i}$ and $\sigma_{i+1} \dots \sigma_n \models \varphi^{>}_{j,\sigma_i,w_i}$.
> - $u \models \varphi^{i}_{w_0}$ and $u \models \varphi^{t}_{w_{n+1}}$.

We want to ensure that the domain of the translation is an $\mathcal{F}$-language. For this we add the following condition of *exhaustiveness*:

For any $\sigma \in \Sigma$, $u, w \in \Sigma^*$, there exists a pair $\left(\varphi^{<}_{j,\sigma,v}, \varphi^{>}_{j,\sigma,v}\right)$ with $j \leq k$ and $v \in S$ such that $u \models \varphi^{<}_{j,\sigma,v}$ and $v \models \varphi^{>}_{j,\sigma,v}$.

Remark 3.2.1. An $\mathcal{F}$-translation $\mathcal{T} = \left(k, S, \left(\varphi^{<}_{j,\sigma,v}, \varphi^{>}_{j,\sigma,v}\right)_{\substack{1\leq j\leq k \\ \sigma\in\Sigma, v\in S}}, \left(\varphi^{i}_{v}\right)_{v\in S}, \left(\varphi^{t}_{v}\right)_{v\in S}\right)$ might not define a function but a relation. In order to ensure that it is functional, we have to add some conditions:

- Let $j, j' \leq k, \sigma \in \Sigma, v_1 \neq v_2 \in S$. Either $\varphi^{<}_{j,\sigma,v_1} \wedge \varphi^{<}_{j',\sigma,v_2}$ is unsatisfiable or $\varphi^{>}_{j,\sigma,v_1} \wedge \varphi^{>}_{j',\sigma,v_2}$ is unsatisfiable.
- Let $v_1 \neq v_2 \in S$, then $\varphi^{i}_{v_1} \wedge \varphi^{i}_{v_2}$ is unsatisfiable.
- Let $v_1 \neq v_2 \in S$, then $\varphi^{t}_{v_1} \wedge \varphi^{t}_{v_2}$ is unsatisfiable.

In the following we will always assume that a translation satisfies these conditions.

Example 3.2.1. We give a translation defining f_{ends}. Let $\Sigma = \{a, b\}$ and $\mathcal{T} = \left(k, S, \left(\varphi^{<}_{j,\sigma,v}, \varphi^{>}_{j,\sigma,v}\right)_{\substack{1\leq j\leq k \\ \sigma\in\Sigma, v\in S}}, \left(\varphi^{i}_{v}\right)_{v\in S}, \left(\varphi^{t}_{v}\right)_{v\in S}\right)$ with $k = 2$ and $S = \{\varepsilon, a\}$.

- $\varphi^{<}_{1,a,\varepsilon} = \exists x\ \mathsf{min}(x) \wedge b(x)$ and $\varphi^{>}_{1,a,\varepsilon} = \top$.
- $\varphi^{<}_{2,a,\varepsilon} = \top$ and $\varphi^{>}_{2,a,\varepsilon} = \exists x\ \mathsf{max}(x) \wedge b(x)$.
- $\varphi^{<}_{1,a,a} = \neg\exists x\ \mathsf{min}(x) \wedge b(x)$ and $\varphi^{>}_{1,a,a} = \neg\exists x\ \mathsf{max}(x) \wedge b(x)$
- $\varphi^{<}_{1,b,\varepsilon} = \neg\exists x\ \mathsf{min}(x) \wedge a(x)$ and $\varphi^{>}_{1,b,\varepsilon} = \top$.
- $\varphi^{<}_{2,b,\varepsilon} = \top$ and $\varphi^{>}_{2,b,\varepsilon} = \neg\exists x\ \mathsf{max}(x) \wedge a(x)$.
- $\varphi^{<}_{1,b,a} = \exists x\ \mathsf{min}(x) \wedge a(x)$ and $\varphi^{>}_{1,b,a} = \exists x\ \mathsf{max}(x) \wedge a(x)$
- $\varphi^{i}_{\varepsilon} = \top$ and $\varphi^{t}_{\varepsilon} = \top$.
- All the other formulas are set to $\bot$.

3.2.1 From machines to logics

Proposition 3.2.1. *Let $\mathbf{V}$ be a monoid variety and let $\mathcal{F}$ be a corresponding logic. Let f be a function definable by a complete $\mathbf{V}$-bimachine. Then f is definable by an $\mathcal{F}$-translation.*

Proof: Let $B = (L, R, \omega, \lambda, \rho)$ be a complete $\mathbf{V}$-bimachine. We can assume that the automata L and R are complete (for instance by taking the automata associated with $\sim_L$ and $\sim_R$, respectively). Let $l \in L$, then there is an $\mathcal{F}$-formula φ_l^L recognizing the language $L_{l_0,l}(L)$. Similarly for $r \in R$ let φ_r^R be an $\mathcal{F}$-formula recognizing $L_{r_0,r}(R)$. Let us define $\mathcal{T} = \left(k, S, \left(\varphi^<_{j,\sigma,v}, \varphi^>_{j,\sigma,v}\right)_{\substack{1\leq j\leq k \\ \sigma\in\Sigma, v\in S}}, \left(\varphi_v^i\right)_{v\in S}, \left(\varphi_v^t\right)_{v\in S}\right)$ with $k = |L|\cdot|R|$, S being the set of outputs in B. For convenience, we will index the formulas with pairs of states instead of integer $\leq |L|\cdot|R|$. Let $l \in L, \sigma \in \Sigma, r \in R, v \in S$ such that $\omega(l, \sigma, r) = v$. We define $\varphi^<_{(l,r),\sigma,v} = \varphi_l^L$ and $\varphi^>_{(l,r),\sigma,v} = \varphi_r^R$. We also define for $v \in S$, $\varphi_v^i = \bigwedge_{\lambda(r)=v} \varphi_r^R$ and $\varphi_v^t = \bigwedge_{\rho(l)=v} \varphi_l^L$. We can see that since L and R are complete and ω is total, the translation is indeed exhaustive.

Let $(u, w) \in [\![B]\!]$. Let $u = \sigma_1 \dots \sigma_n \in \Sigma^n$, let $l_0 \xrightarrow{\sigma_1} l_1 \cdots l_{n-1} \xrightarrow{\sigma_n} l_n$ be the execution of L on u and let $r_n \xleftarrow{\sigma_1} r_{n-1} \cdots r_1 \xleftarrow{\sigma_n} r_0$ be the execution of R on u. Let $w = v_0 v_1 \dots v_n v_{n+1}$ such that $\omega(l_{i-1}, \sigma_i, r_{n-i}) = v_i$ for $1 \leq i \leq n$, $\lambda(r_n) = v_0$ and $\rho(l_n) = v_{n+1}$. Then for $1 \leq i \leq n$ we have both $\sigma_1 \dots \sigma_{i-1} \models \varphi^<_{(l_{i-1},r_{n-i}),\sigma_i,v_i}$ and $\sigma_{i+1} \dots \sigma_n \models \varphi^>_{(l_{i-1},r_{n-i}),\sigma_i,v_i}$. We also have $u \models \varphi^i_{v_0}$ and $u \models \varphi^t_{v_{n+1}}$. Hence $(u, w) \in [\![\mathcal{T}]\!]$. Conversely let us assume $(u, w) \in [\![\mathcal{T}]\!]$. With the same notations as above, we have for $1 \leq i \leq n$, $\sigma_1 \dots \sigma_{i-1} \models \varphi^<_{(l_{i-1},r_{n-i}),\sigma_i,v_i}$ and $\sigma_{i+1} \dots \sigma_n \models \varphi^>_{(l_{i-1},r_{n-i}),\sigma_i,v_i}$. We also have $u \models \varphi^i_{v_0}$ and $u \models \varphi^t_{v_{n+1}}$. We just have to remark that the sequence of left (resp. right) states corresponds to an execution on L (resp. R). Then we have automatically that $v_i = \omega(l_{i-1}, \sigma_i, r_{n-i})$, $\lambda(r_n) = v_0$ and $\rho(l_n) = v_{n+1}$. Thus we obtain $(u, w) \in [\![B]\!]$. □

3.2.2 From logics to machines

Proposition 3.2.2. *Let $\boldsymbol{V}$ be a monoid variety and let $\mathcal{F}$ be a corresponding logic. Let f be a function definable by an $\mathcal{F}$-translation. Then f is definable by a complete $\boldsymbol{V}$-bimachine.*

Proof: Let $\mathcal{T} = \left(k, S, \left(\varphi^<_{j,\sigma,v}, \varphi^>_{j,\sigma,v}\right)_{\substack{1\leq j\leq k \\ \sigma\in\Sigma, v\in S}}, \left(\varphi_v^i\right)_{v\in S}, \left(\varphi_v^t\right)_{v\in S}\right)$ be an $\mathcal{F}$-translation.

We will describe a bimachine $B = (L, R, \omega, \lambda, \rho)$ equivalent to $\mathcal{T}$. We define two sets of formulas: $F_L = \left\{\varphi^<_{j,\sigma,v} \mid 1 \leq j \leq k,\ \sigma \in \Sigma,\ v \in S\right\} \cup \left\{\varphi^t_v \mid v \in S\right\}$ and $F_R = \left\{\varphi^>_{j,\sigma,v} \mid 1 \leq j \leq k,\ \sigma \in \Sigma,\ v \in S\right\} \cup \left\{\varphi^i_v \mid v \in S\right\}$. For any formula in F_L there is a $\mathbf{V}$-congruence recognizing the same language. Let $\sim_L$ be a $\mathbf{V}$-congruence finer than all of these. Similarly we can define $\sim_R$ and we obtain naturally the automata L, R. Then for $[u] \in L$, $[w] \in R$ and $\sigma \in \Sigma$, $\omega([u], \sigma, [w]) = v$ with v being the unique

word in S such that for some $j \leq k$, $u \models \varphi^{<}_{j,\sigma,v}$ and $w \models \varphi^{>}_{j,\sigma,v}$. Let us remark that ω is total by exhaustiveness. Hence we obtain a complete $\mathbf{V}$-bimachine and exactly as above, we can show it defines f. $\square$

Remark 3.2.2. If $\mathcal{F}$ is a logic corresponding to a variety $\mathbf{V}$, it can easily be seen that $\mathcal{F}$-translations with the formulas $\varphi^{>}_{j,\sigma,v} = \top$ are equivalent to $\mathbf{V}$-DFTs. Moreover, if we set for $v \neq \varepsilon$ that $\varphi^{t}_{v} = \bot$, we obtain the functions definable by $\mathbf{V}$-DFTs with no terminal output.

Chapter 4

Algebraic characterization of classes of subsequential functions

The first part of this chapter deals with a minimization procedure described by [Cho03] and we show it preserves the equalities of the transition congruence. In particular, one can decide if a subsequential function is definable by an aperiodic DFT. In the second part we show that a determinization algorithm of transducers preserves the aperiodicity of the transition monoid.

4.1 Minimization

In order to have an algebraic classification of functions, we need a notion similar to the one of minimal automaton. In the case of subsequential functions, there is such notion which we call the minimal DFT. We describe the construction of a minimal DFT given in [Cho03] and prove that it preserves equality in the transition congruence. However we do not prove here that it is correct and refer the reader to the original paper. This minimization is canonical and not machine dependent.

Let f be a subsequential function definable by some DFT $T = (Q, q_0, F, \delta, i, t)$. For $u \in \Sigma^*$, we define when it exists $\widehat{f}(u) = \wedge \{f(uw) \mid uw \in \mathrm{dom}(f)\}$. We define the *syntactic congruence* of f, for $u, v \in \Sigma^*$, $u \sim_f v$ if:

For all $w \in \Sigma^*$, $uw \in \mathrm{dom}(f) \Leftrightarrow vw \in \mathrm{dom}(f)$ and when it is defined, $\widehat{f}(u)^{-1} f(uw) = \widehat{f}(v)^{-1} f(vw)$.

Then the transducer $T_f = (Q_f, q_{0f}, F_f, \delta_f, i_f, t_f)$ is equivalent to T with:

- $Q_f = \left\{ [u] \mid u \in \mathrm{dom}(\widehat{f}) \right\}$

- $q_{0f} = [\varepsilon]$ (we assume that the domain of T is not empty).
- $\delta([u], \sigma) = ([u\sigma], \widehat{f}(u)^{-1}\widehat{f}(u\sigma))$ when $\widehat{f}(u\sigma)$ is defined.
- $F_f = \{[u] \mid u \in \mathrm{dom}(f)\}$.
- $i_f([\varepsilon]) = \widehat{f}(\varepsilon)$.
- $t_f([u]) = \widehat{f}(u)^{-1} f(u)$ whenever $[u] \in F_f$.

Proposition 4.1.1. *Given a monoid variety $\mathbf{V}$, let f be a subsequential function. The function f is definable by a $\mathbf{V}$-DFT if and only if its minimal transducer is a $\mathbf{V}$-DFT.*

Proof: According to Proposition 2.2.1 we only have to show $\equiv_T \sqsubseteq \equiv_{T_f}$. Let f be a subsequential function definable by some DFT $T = (Q, q_0, F, \delta, i, t)$. For $q \in Q$ let $T_q = (Q, q, F, \delta, \varepsilon, t)$ and let f_q be the corresponding function. Let $m_1, m_2 \in \Sigma^*$ two words such that $m_1 \equiv_T m_2$. We construct T_f using the minimization in [Cho03]. Let v be a word in Σ^*. Let $q_0 \xrightarrow{vm_1|x_1} p$ (if p does not exist, we can add a garbage state in T). Then obviously $q_0 \xrightarrow{vm_2|x_2} p$ and for all $w \in \Sigma^*$, $vm_1w \in \mathrm{dom}(f) \Leftrightarrow vm_2w \in \mathrm{dom}(f)$. Then when it is defined:

$$\widehat{f}(vm_1) = ix_1 \bigwedge \{f_p(w) \mid w \in \mathrm{dom}(f_p)\}$$

$$\widehat{f}(vm_2) = ix_2 \bigwedge \{f_p(w) \mid w \in \mathrm{dom}(f_p)\}$$

Let $s = \bigwedge \{f_p(w) \mid w \in \mathrm{dom}(f_p)\}$, then for any $w \in \Sigma^*$ such that $vm_1w \in \mathrm{dom}(f)$:

$$\widehat{f}(vm_1)^{-1} f(vm_1w) = s^{-1}x_1^{-1}i^{-1}ix_1f_p(w) = s^{-1}f_p(w)$$

$$\widehat{f}(vm_2)^{-1} f(vm_2w) = s^{-1}x_2^{-1}i^{-1}ix_2f_p(w) = s^{-1}f_p(w)$$

Hence $vm_1 \sim_f vm_2$ for any $v \in \Sigma^*$. We have in T_f for any $v \in \Sigma^*$: $[v] \xrightarrow{m_1} [vm_1] = [vm_2]$ and $[v] \xrightarrow{m_2} [vm_2]$. Hence $m_1 \equiv_{T_f} m_2$ and $\equiv_{T_f}$ is the coarsest amongst the transition congruences of all the DFTs defining f. □

Theorem 4.1.1. *Let $\mathbf{V}$ be a monoid variety. Let us assume that the membership problem is decidable for $\mathbf{V}$. Then there is an algorithm to decide if a function defined by a DFT is definable by a $\mathbf{V}$-DFT.*

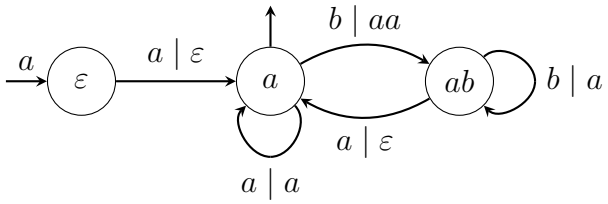


Figure 4.1: Minimal DFT of g.

Proof: This follows easily from Proposition 4.1.1: We only have to construct the minimal DFT and check if its transition monoid belongs to $\mathbf{V}$. □

Example 4.1.1. Let us consider the sequential function g which is equal to f_{ends}, restricted to L_{ends}. In other words, g_{ends} is defined on words that start and end with an a and replaces each letter of the input by an a. We have:

- $\widehat{g}(\varepsilon) = a$.
- For $w \in a + a\Sigma^* a$, $\widehat{g}(w) = a^{|w|}$.
- For $w \in a\Sigma^* b$, $\widehat{g}(w) = a^{|w|+1}$.
- For $w \in b\Sigma^*$, the function is not defined.

Hence we obtain the following congruence classes: $[\varepsilon]$, $[a] = [a + a\Sigma^*]$, $[ab] = [a\Sigma^* b]$ and $[b] = [b\Sigma^*]$. From this we obtain the minimal DFT of g given in Figure 4.1, with states labelled as representatives of congruence classes.

4.2 Determinization of aperiodic transducers

It has been shown that it is decidable whether an NFT is determinizable or not. In this case we will describe the determinization algorithm given in [BC02] and show it preserves the aperiodicity of the transducer. In particular, we have that a subsequential function is definable by an aperiodic DFT if and only if it is definable by an aperiodic NFT, which does not hold for a general variety as can be seen in Section 6.2.

Let $T = (Q, I, F, \Delta, i, t)$ be an NFT. We give a construction of $T' = (Q', S_0, F', \delta, i', t')$ equivalent to T. Since we are only interested in the transitions of T', we will not describe the final states nor the terminal output function.

In the obtained transducer T', a state S is a set of pairs $(p, w) \in Q \times \Sigma^*$. Let $X \subseteq \Sigma^*$, $X \neq \varnothing$ then we define $\wedge X$ to be the longest common prefix of all the words in X. Let $i' = \wedge\{i(q) \mid q \in I\}$. Let $S_0 = \{(q, w) \mid i(q) = i'w\}$. Then for S_1 a state and $\sigma \in \Sigma$, we define $R_2 = \left\{(q, vu) \mid (p, v) \in S_1 \text{ and } p \xrightarrow{\sigma|u} q\right\}$. Let $s = \wedge\{w \mid (q, w) \in R_2\}$. Then we define $S_2 = \{(q, w) \mid (q, sw) \in R_2\}$ and there is a transition $S_1 \xrightarrow{\sigma|s} S_2$ in T'. Let us remark that, by construction, in the underlying automaton of T', $S_1 \xrightarrow{u} S_2$ implies that the states of S_2 are exactly the states reachable by reading u from a state of S_1.

Proposition 4.2.1. *Given a subsequential function defined by an* $\boldsymbol{A}$*-NFT, the transducer obtained by determinization is an* $\boldsymbol{A}$*-DFT.*

Proof: First let us denote by $u \prec v$ that u is a suffix of v. And let $u \approx v$ if $u \prec v$ or $v \prec u$.
Let f be a rational function defined by an $\mathbf{A}$-NFT $T = (Q, I, F, \Delta, i, t)$. There exists an integer n such that in the underlying automaton of T, for all $p, q \in Q$, we have $p \xrightarrow{u^n} q$ if and only if $p \xrightarrow{u^{n+1}} q$.

An automaton is *counter-free* if for any state q, any word u, any positive integer k, $q \xrightarrow{u^k} q \Rightarrow q \xrightarrow{u} q$. Since T' is deterministic, we only have to show that it is counter-free (see *e.g.* [DG08]).

Let u be a word, let R_0 be a state of T' and let k be, if it exists, the smallest positive integer such that $R_0 \xrightarrow{u^k} R_0$. Let $R_0 \xrightarrow{u|x_1} R_1 \ldots, R_{k-1} \xrightarrow{u|x_0} R_0$ be the corresponding sequence of transitions in T'. We have to show that $k = 1$. Let us remark that all the states of the R_j's must be the same since they are the set of states reachable, in T, by u^n from the states of R_0. Let $R_j = \{(q_1, v_{1,j}), \ldots, (q_m, v_{m,j})\}$ be pairwise distinct for $j \in \{0, \ldots, k-1\}$. We assume that not all x_j's are empty words, otherwise, the conclusion is immediate.

Let $q_{i_1} \xrightarrow{u|x_{i_1,i_2}} q_{i_2}$ denote the transitions in T. Then by construction of T', we have for any for $j \in \{0, \ldots, k-1\}$:

$$v_{i_1,j} x_{i_1,i_2} = x_{j+1} v_{i_2,j+1}$$

Let $i_0, i_1, \ldots, i_t$ be an index sequence such that $q_{i_l} \xrightarrow{u} q_{i_{l+1}}$ for $l \in \{0, \ldots, t-1\}$ and t a multiple of k. Then we obtain for any $j, j' \in \{0, \ldots, k-1\}$:

$$\begin{array}{ll} v_{i_0,j} & x_{i_0,i_1} \cdots x_{i_{t-1},i_t} = x_{j+1} \cdots x_{j+t} v_{i_t,j} \\ v_{i_0,j'} & x_{i_0,i_1} \cdots x_{i_{t-1},i_t} = x_{j'+1} \cdots x_{j'+t} v_{i_t,j'} \end{array}$$

In particular the two words have a common suffix which means that $v_{i_t,j}$ and $v_{i_t,j'}$ have a common suffix. This suffix can be arbitrarily large since t can be chosen arbitrarily large. Hence $v_{i,j} \approx v_{i,j'}$ for any i, j, j' (any state q_i is reachable by a sufficiently long sequence).

Let $X_0 = x_0 \cdots x_{k-1}$. For $j \in \{1, \ldots, k-1\}$ and let $X_j = x_j \cdots, x_{j-1}$ denote the corresponding rotations of X_0. Let us remark that $X_j x_j = x_j X_{j+1}$. Since T' is finite, there exists an index i such that we have both: $q_i \xrightarrow{u^t} q_i$ and $q_i \xrightarrow{u^{t+1}} q_i$, and let $i = i_1, i_2, \ldots, i_t, i$ and $i = i'_1, i'_2, \ldots, i'_{t+1}, i$ be the corresponding index sequences. By aperiodicity, we can choose $t = sk$, a multiple of k. Let $Y = x_{i_1,i_2} \cdots x_{i_t,i_1}$ and let $Y' = x_{i'_1,i'_2} \cdots x_{i'_{t+1},i'_1}$. Then we have for any j:

$$\begin{aligned} v_{i,j}Y &= (X_j)^s v_{i,j} \\ v_{i,j+1}Y &= (X_{j+1})^s v_{i,j+1} \\ v_{i,j}Y' &= (X_j)^s x_j v_{i,j+1} \end{aligned}$$

Let us assume that $v_{i,j} \prec v_{i,j+1}$ which means that $v_{i,j+1} = w v_{i,j}$.

$$\begin{aligned} v_{i,j}Y &= (X_j)^s v_{i,j} && (\alpha) \\ w v_{i,j}Y &= (X_{j+1})^s w v_{i,j} && (\beta) \\ v_{i,j}Y' &= (X_j)^s x_j w v_{i,j} && (\gamma) \end{aligned}$$

From (α) and (β), we have: $w(X_j)^s = (X_{j+1})^s w$. And by multiplying by x_j and using $X_j x_j = x_j X_{j+1}$ we have: $x_j w (X_j)^s = (X_j)^s x_j w$.

From (γ), by multiplying by Y' on the right k times we have: $v_{i,j}(Y')^k = ((X_j)^s x_j w)^k v_{i,j}$. Moreover, since $(Y')^k$ corresponds to a sequence of length a multiple of k, we also have: $v_{i,j}(Y')^k = (X_j)^{ks+1} v_{i,j}$. Hence $(X_j)^{ks+1} = ((X_j)^s x_j w)^k$. And by combining the two we obtain $(X_j)^{ks+1} = (X_j)^{ks}(x_j w)^k$, hence:

$$X_j = (x_j w)^k$$

This yields

$$\begin{aligned} (X_{j+1})^s w &= w(X_j)^s \\ &= w(x_j w)^{ks} \\ &= (w x_j)^{ks} w \end{aligned}$$

Hence

$$X_{j+1} = (w x_j)^k$$

Let q_l be another state such that $q_l \xrightarrow{u^t} q_l$. If $v_{l,j} \prec v_{l,j+1}$ then $v_{l,j+1} = w_l v_{l,j}$. However, it also implies that $(x_j w)^k = (x_j w_l)^k$, hence $w_l = w$. On the other hand, let us assume $v_{l,j} \succ v_{l,j+1}$ with $w_l v_{l,j+1} = v_{l,j}$. Let Y_l and Y'_l correspond this time to sequences associated with $q_l \xrightarrow{u^t} q_l$ and $q_l \xrightarrow{u^{t-1}} q_l$, respectively.

$$\begin{aligned} w_l v_{l,j+1} Y_l &= (X_j)^s w_l v_{l,j+1} \\ v_{l,j+1} Y_l &= (X_{j+1})^s v_{l,j+1} \\ v_{l,j+1} Y'_l &= (X_{j+1})^s x_{j+1} \cdots x_{j-1} w_l v_{l,j+1} \end{aligned}$$

As above, the first two equations give: $(X_j)^s w_l = w_l (X_{j+1})^s$

From the third equation we have: $((X_{j+1})^s x_{j+1} \cdots x_{j-1} w_l)^k = (X_{j+1})^{ks+k-1}$. Let us remark that $(X_{j+1})^s x_{j+1} \cdots x_{j-1} = x_{j+1} \cdots x_{j-1} (X_j)^s$, hence $(X_{j+1})^s x_{j+1} \cdots x_{j-1} w_l = x_{j+1} \cdots x_{j-1} w_l (X_{j+1})^s$. From this we obtain $(x_{j+1} \cdots x_{j-1} w_l)^k = (X_{j+1})^{k-1}$.

Now from above we can write:

$$\begin{aligned} (x_{j+1} \cdots x_{j-1} w_l)^k &= ((w x_j)^k)^{k-1} \\ x_{j+1} \cdots x_{j-1} w_l &= (w x_j)^{k-1} \\ w x_j x_{j+1} \cdots x_{j-1} w_l &= (w x_j)^k \\ w X_j w_l &= X_{j+1} \end{aligned}$$

Hence by length, we obtain $w = w_l = \varepsilon$.

Let q_l be a state reachable from q_i (any state is reachable from such a looping state). Then it can be reached by a sequence of length t and there is a correspond sequence of outputs denoted by Z_l such that:

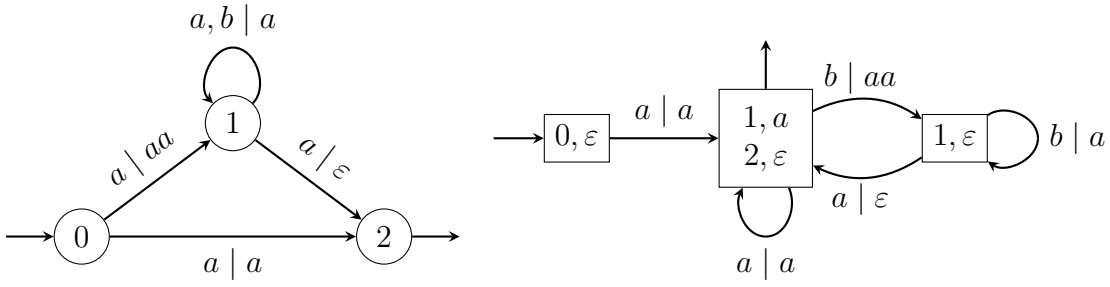
$$\begin{aligned} v_{i,j} Z_l &= (X_j)^s v_{l,j} \\ w v_{i,j} Z_l &= (X_{j+1})^s v_{i,j+1} \end{aligned}$$

Hence $w v_{l,j} = v_{l,j+1}$.

Finally, w is a common prefix to all $v_{i',j+1}$ (j fixed and $i' \in \{1, \cdots, m\}$) and must by definition be empty. The reasoning is the same if we assume $v_{i,j} \succ v_{i,j+1}$. Hence $v_{i',j} = v_{i',j+1}$ and therefore $k = 1$, which means that T' is counter-free and thus is aperiodic. □

Remark 4.2.1. In Section 6.2 we give several examples of varieties $\mathbf{V}$ for which, contrary to the variety of aperiodic monoids, determinization of a transducer does not preserve the variety of the transition monoid. In fact we even give examples of subsequential functions which are not definable by any $\mathbf{V}$-DFT yet are definable by a $\mathbf{V}$-NFT.

Example 4.2.1. Let g be defined by the aperiodic NFT of Figure 4.2 (a). Then the DFT obtained by determinization is the one of Figure 4.2 (b).



(a) NFT defining g.

(b) Determinized transducer.

Figure 4.2: Determinization of a transducer

Chapter 5

Algebraic characterization of classes of rational functions

In this chapter we give a characterization of functions that can be defined by a **V**-NFT for a given variety **V**. The goal is ultimately to decide if a function can be defined by a logical translation for a logic corresponding to the variety. The notion of minimal automaton does not transfer to NFTs. However there is some notion of canonical bimachine, introduced in [RS91], which is the foundation of the algebraic characterization we give.

5.1 V-transducer and V-bimachines

Since the algebraic characterization deals with bimachines, we need to define a notion of bimachine equivalent to **V**-transducers. A bimachine is called a ***V**-bimachine* if both its automata are **V**-automata. We show the following relations between **V**-transducers and **V**-bimachines.

Proposition 5.1.1. *Let **V** be a monoid variety. Let f be a rational function. Then there is a equivalence between the three following properties:*

1. *The function f is definable by an unambiguous **V**-transducer.*
2. *The function f is definable by a complete **V**-bimachine.*
3. *The function f is definable by a **V**-bimachine and $\mathrm{dom}(f)$ is a **V**-language.*

Furthermore there is a construction from each one to the others.

Proof: 1. $\Rightarrow$ 3. follows from the construction given in Section 3.1.2. Since $\equiv_T$ is a $\mathbf{V}$ congruence, we have that L and R are both $\mathbf{V}$-automata. Moreover, $\mathrm{dom}(f)$ is a $\mathbf{V}$-language since it is accepted by the underlying automaton of a $\mathbf{V}$-NFT.

3. $\Rightarrow$ 2. follows from the construction given in Proposition 3.1.1. We can choose the automaton A to be a $\mathbf{V}$-automaton since the domain is a $\mathbf{V}$-language. Hence we obtain a complete $\mathbf{V}$-bimachine.

2. $\Rightarrow$ 1. follows from the construction given in Section 3.1.1. Since the bimachine is complete, the underlying automaton of the obtained transducer is just the product $L \times \tilde{R}$, with $\tilde{R}$ being the left automaton obtained by reversing the transitions in R. We have that $\equiv_{\tilde{R}}$ and $\equiv_R$ are equal. Therefore the transducer we obtain is a $\mathbf{V}$-transducer. □

Remark 5.1.1. For some variety $\mathbf{V}$, there exist functions definable by a $\mathbf{V}$-bimachine for which the domain is *not* a $\mathbf{V}$-language. An example is given in Section 6.3.

A question that remains is the following one: If a function is definable by a $\mathbf{V}$-NFT, is it definable by some unambiguous $\mathbf{V}$-NFT ? In the next part, we will see that the answer is yes in the particular case of aperiodic transducers.

5.2 Canonical bimachine

In this part we give the construction, for a given rational function, of a canonical bimachine that is described in [RS91] and several properties of this bimachine. It is canonical in the sense that it is not machine dependant and only depends on the function itself. Although, we give the construction of the bimachine, we refer the reader to the original paper for a proof of correctness.

In order to define the bimachine we define a left congruence from which we define the right automaton. Then from this we define the right congruence which yields the left automaton.

5.2.1 Left congruence

Let f be a rational function on Σ.

Let the *left distance* between two words u, v be $\|u, v\| = |u| + |v| - 2|u \wedge v|$ where $u \wedge v$ is the longest common prefix of u and v.

We define the *left syntactic congruence* of f (which is a left congruence) by

$u \sim_{R_0} v$ if:

$$\forall w \in \Sigma^*,\ wu \in \mathrm{dom}(f) \Leftrightarrow wv \in \mathrm{dom}(f)$$
and
$$\sup_{w\in\Sigma^*} \{\|f(wu), f(wv)\|\} < \infty$$

Let $\sim_R$ be a left congruence finer than $\sim_{R_0}$. Let $R = \Sigma^*/\sim_R$, $R_0 = \Sigma^*/\sim_{R_0}$ and let $r_0 = [\varepsilon]$ be the initial element of R.

For $r, r' \in R$, $\sigma \in \Sigma$, if $r' \xleftarrow{\sigma} r$ then for convenience we denote by $r' = \sigma r$ the action of Σ on R.

For this fixed R we construct a bimachine $B^R = (L^R, R, \omega^R, \lambda^R, \rho^R)$ which defines f according to [RS91]. By taking R_0 as a right automaton we get a completely canonical bimachine $B^0 = (L^0, R_0, \omega^0, \lambda^0, \rho^0)$. Since we are only interested in the automata of the bimachine, we will not give the descriptions of ω^R, λ^R and ρ^R, and once again we refer the reader to [RS91].

Remark 5.2.1. The construction is completely symmetrical: we can define similarly a right congruence: $u \sim_{L_0} v$ if the supremum of the *right distance* between $f(uw)$ and $f(vw)$ over all words w is less than infinity. Then for any right congruence $\sim_L$ finer than $\sim_{L_0}$, we can define $B^L = (L, R^L, \omega^L, \lambda^L, \rho^L)$ defining f. We denote R^{L_0} by R^0 to simplify.

First we show that the automaton R_0 (as well as L_0) has some minimal property, *i.e.* its transition congruence is coarser than the transition congruence of any transducer defining f.

Proposition 5.2.1. *Let f be a rational function defined by some **V**-NFT. Then the automaton $R_0 = (\Sigma^*/\sim_{R_0}, [\varepsilon], \delta)$, where δ is the natural left action on $\Sigma^*/\sim_{R_0}$, is a **V**-automaton.*

Proof: Let $T = (Q, I, F, \Delta, i, t)$ be a **V**-NFT defining f. According to Proposition 2.2.1, it suffices to prove $\sim_T \sqsubseteq \equiv_{R_0}$. Let $w \in \Sigma^*$ be two words, let $m_1 \equiv_T m_2$ and let $q_0 \xrightarrow{w|x} p \xrightarrow{m_1|y} q_f$ be an accepting run of T on wm_1. Then there is an accepting run $q_0 \xrightarrow{w|x} p \xrightarrow{m_2|y'} q_f$. Thus the distance $\|f(wm_1), f(wm_2)\| \leq k(|m_1| + |m_2| + 1)$ with k being the maximum length of an output in T. Hence $m_1 \sim_{R_0} m_2$. □

5.2.2 Right congruence

Now that we have defined the right automaton of the bimachine, we need to define the left one. In order to do that we define this time a right congruence.

Let $r \in R$, we define the function

$$\widehat{f_r}(u) = \bigwedge \{f(uv) \mid vr_0 = r\}$$

These functions are similar to the prefix function defined for minimizing subsequential functions in Section 4.1. However this time we need a finite family of prefix functions. It is proven in [RS91] that $\widehat{f_{\sigma r}}(u)$ is a prefix of $\widehat{f_r}(u\sigma)$ for $\sigma \in \Sigma, u \in \Sigma^*$ and $r \in R$. Then for $u, v \in \Sigma^*$ we say $u \sim_{L^R} v$ if for any $\sigma \in \Sigma, w \in \Sigma^*$ and $r \in R$ we have when it is defined both:

$$\begin{aligned} \widehat{f_{\sigma r}}(uw)^{-1}\widehat{f_r}(uw\sigma) &= \widehat{f_{\sigma r}}(vw)^{-1}\widehat{f_r}(vw\sigma) \\ \widehat{f_{r_0}}(uw)^{-1}f(uw) &= \widehat{f_{r_0}}(vw)^{-1}f(vw) \end{aligned}$$

This right congruence has finite index, according to [RS91], thus we obtain the left automaton of the bimachine B^R defining f.

For the purpose of this report, we want to study the algebraic properties of L^R and for this we need to show that the functions $\widehat{f_r}$, $r \in R$ maintain some algebraic properties of T (given below in Proposition 5.2.2 and Corollary 5.2.1) and for this we give the construction of a family of transducers defining them. Let us give a construction of a transducer defining $\widehat{f_r}$, for $r \in R$. According to [RS91] there exists $L_r = \{v_1, \dots v_m\}$ finite such that $\widehat{f_r}(u) = \bigwedge \{f(uv) \mid v \in L_r\}$.

Let $T \times R = (Q \times R, I \times R, F \times \{r_0\}, \Delta', i', t')$ be the transducer that does exactly as T does except that it guesses in a non-deterministic fashion the congruence class $\sim_R$ of the word read, and checks by taking the transitions of R backwards to reach r_0. In particular this transducer defines the function f, the states are just refined to also give the class in R of the suffix left to read.

Now from this transducer we want to build $T_r = (Q_R, I_R, F_r, \Delta_R, i_R, t_r)$ defining $\widehat{f_r}$ with $i_R = \wedge \{i(q) \mid q \in I\}$. The construction will be similar to the determinization algorithm, except that only states with same class in R will be merged. The algorithm will end since for $w_1 \sim_R w_2$ we have by definition the left distance between $f(ww_1)$ and $f(ww_2)$ that is bounded.

Each state of T_r is composed of a set of pairs $(q, w) \in Q \times \Sigma^*$ and a class $s \in R$. For $i_R = \wedge \{i(q) \mid q \in I\}$ and $s \in R$, each $(\{(q, w) \mid i(q) = i_R w\}, s)$ belongs to I_R. Let (P, s) be a state of Q_R, let $\sigma \in \Sigma$ and $s' \in \Sigma$ such that $s = \sigma s'$. Then we can determine a set

$$S = \left\{(q', wu) \mid \exists (q, w) \in P \text{ and } q \xrightarrow{\sigma|u} q'\right\}$$

Let v be the longest common prefix of the words of S then we define

$$P' = \{(q', w') \mid (q', vw') \in S\}$$

There is then a transition $(P,s) \xrightarrow{\sigma|v} (P',s')$. We define $F_r = \{(P,s) \in Q_R \mid s = r\}$. Let $(P,r) \in F_r$ with $P = \{(q_1,w_1),\dots,(q_m,w_m)\}$. For any $v \in L_r$ there exists j_v such that for some $q_{f_v} \in F$, $q_{j_v} \xrightarrow{v|u_v} q_{f_v}$. Then we define $t_r((P,r)) = \bigwedge \{w_{j_v}u_v \mid v \in L_r\}$.

By construction, T_r defines $\widehat{f_r}$, furthermore this construction has the advantage of yielding an unambiguous transducer: for a given word, there is only one sequence of states in R corresponding to it and since all states in $Q \times R$ following this sequence are merged, there is only one possible accepting run.

Let $T_R = (Q_R, I_R, \tilde{\Delta}_R)$ denote the (unique) underlying automaton associated with the T_r's (when not taking final states into account).

For the following, we also need to show that f is definable by a NFT with T_R as its underlying automaton. We define $T_f = (Q_R, I_R, F_f, \Delta_R, i_R, t_f)$ with the final states $F_f\,\{(P,r_0) \mid \exists (p,w) \in P \text{ and } p \in F\}$ and the terminal function $T_f(P,r_0) = wt(p)$ for $(p,w) \in P$ and $p \in F$.

The construction of T_R allows us to claim the following property and its corollary.

Proposition 5.2.2. *Let f be a rational function defined by some **V**-NFT. If the automaton T_R is a **V**-automaton, then so is the automaton $L^R = (\Sigma^*/\sim_{L^R}, [\varepsilon], \delta)$, where δ is the natural right action on $\Sigma^*/\sim_L$.*

Proof: Let $T = (Q, I, F, \Delta, i, t)$ be a **V**-NFT defining f. According to Proposition 2.2.1 it suffices to show $\equiv_{T^R} \sqsubseteq \sim_{L^R}$. Let $m_1 \equiv_{T_R} m_2$, $r \in R$ and $w \in \Sigma^*$. Let $(P_0, m_1 w\sigma r) \xrightarrow{m_1|u_1} (P_1, w\sigma r) \xrightarrow{w|v} (P_2, \sigma r) \xrightarrow{\sigma|v'} (P_3, r)$ and $(P_0, m_1 w\sigma r) \xrightarrow{m_2|u_2} (P_1, w\sigma r) \xrightarrow{w|v} (P_2, \sigma r) \xrightarrow{\sigma|v'} (P_3, r)$ be successful runs of T_r (if they exist). Then we have:

$$\begin{aligned}
\widehat{f_r}(m_1 w\sigma) &= u_1 v v' t_r(P_3, r)\\
\widehat{f_r}(m_2 w\sigma) &= u_2 v v' t_r(P_3, r)\\
\widehat{f_{\sigma r}}(m_1 w) &= u_1 v t_{\sigma r}(P_2, \sigma r)\\
\widehat{f_{\sigma r}}(m_2 w) &= u_2 v t_{\sigma r}(P_2, \sigma r)
\end{aligned}$$

By definition $t_{\sigma r}(P_2, \sigma r)$ is indeed a prefix of $v' t_r(P_3, r)$, hence $\widehat{f_{wr}}(m_1)^{-1}\widehat{f_r}(m_1 w) = \widehat{f_{wr}}(m_2)^{-1}\widehat{f_r}(m_2 w)$.

We have $m_1 \in \mathrm{dom}(f)$ if and only if $m_2 \in \mathrm{dom}(f)$. Let $(P_0, m_1 w r_0) \xrightarrow{m_1|u_1} (P_1, w r_0) \xrightarrow{w|v} (P_2, r)$ and $(P_0, m_1 w r_0) \xrightarrow{m_2|u_2} (P_1, w r_0) \xrightarrow{w|v} (P_2, r)$ be successful runs

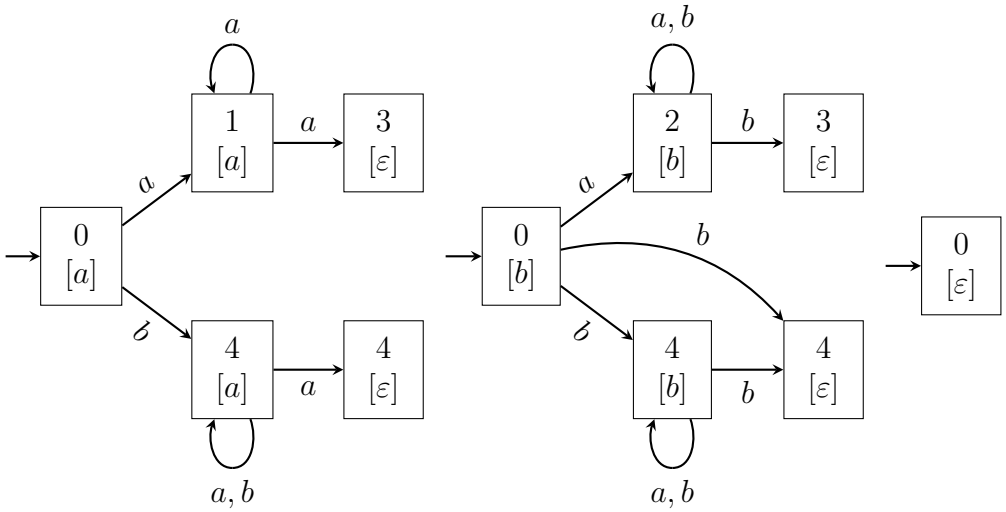


Figure 5.1: Automaton T_R.

of T_f (if they exist). Then we have:

$$\begin{aligned} f(m_1 w) &= u_1 v t_f(P_2, r_0) \\ f(m_2 w) &= u_2 v t_f(P_2, r_0) \\ \widehat{f}_{r_0}(m_1 w) &= u_1 v t_{r_0}(P_2, r_0) \\ \widehat{f}_{r_0}(m_2 w) &= u_2 v t_{r_0}(P_2, r_0) \end{aligned}$$

By definition $t_{r_0}(P_2, r_0)$ is indeed a prefix of $t_f(P_2, r_0)$ since $\varepsilon \in r_0$, hence $\widehat{f}_{r_0}(m_1)^{-1} f(m_1) = \widehat{f}_{r_0}(m_2)^{-1} f(m_2)$, and $m_1 \sim_{L^R} m_2$. □

Corollary 5.2.1. *Let $\boldsymbol{V}$ be a monoid variety stable by determinization of transducers. Let f be definable by a $\boldsymbol{V}$-transducer. Then B^0 is a $\boldsymbol{V}$-bimachine.*

Proof: According to Proposition 5.2.1, R_0 is a $\mathbf{V}$-automaton. Hence by construction T_{R_0} is a $\mathbf{V}$-automaton since $\mathbf{V}$ is stable by determinization. Hence, according to Proposition 5.2.2, L^0 is also a $\mathbf{V}$-automaton. □

Example 5.2.1. We consider the function f_{ends} and one can show that the automaton R_0 is equal to the one of Example 3.1.1, *i.e.* $u \sim_{R_0} v$ if they have the same last letter. The automaton T_R we obtain for the NFT defining f_{ends} is given in Figure 5.1.

5.2.3 Aperiodic case

In Chapter 4 we have proven that aperiodic transducers are stable by determinization. Hence we can decide if a given function is definable by an aperiodic transducer, and claim the following Theorem:

Theorem 5.2.1. *There is an algorithm to decide if a function defined by a transducer is definable by an FO-translation.*

Proof: Let us assume f is definable by an **A**-NFT. Then its domain is an **A**-language and, according to Corollary 5.2.1, the canonical bimachine is an **A**-bimachine. Let us assume that f is definable by an FO-translation. According to Proposition 3.2.2, we can construct a complete aperiodic bimachine defining f. Then according to Proposition 5.1.1 we can construct an **A**-NFT defining f. Hence to decide if a rational function given by a NFT is definable by a **A**-NFT, we need to:

1. Check that the domain is an aperiodic language.
2. Construct the canonical bimachine B^0.
3. If the domain is an aperiodic language and B^0 is aperiodic, from Proposition 5.1.1 we can build a complete aperiodic bimachine B defining the function.
4. From B, according to Proposition 3.2.1, we can construct an FO-translation defining the function.

□

Remark 5.2.2. In particular, a function is definable by an aperiodic transducer if and only if it is definable by an unambiguous transducer, since the construction from bimachine to transducer always yields an unambiguous transducer.

5.3 General variety

In this section we try to generalize the previous result to any variety. We show that if a function is definable by an unambiguous **V**-NFT, then there exists a bimachine amongst a finite set of "canonical" bimachines that is a **V**-bimachine. We first show the result on total functions and then extend it to the general case.

Let A, A' be two deterministic left (resp. right) automata. In order to simplify the notations, we will write $A \sqsubseteq A'$ and say that the automaton A is finer than A' whenever $\sim_A \sqsubseteq \sim_{A'}$.

5.3.1 Total functions

The construction is a finer version of the previous one and heavily uses the following Theorem of [RS91].

Theorem 5.3.1. *Let f be a total rational function, let R be a right automaton finer than R_0 and let L^R be the left automaton of the bimachine B^R. Then we say that L^R is* minimal for R *in the following sense: for any bimachine defining f with R as its right automaton and L' as its left automaton, we have*

$$L' \sqsubseteq L^R$$

We also use the following Proposition which holds for total functions only.

Proposition 5.3.1. *Let $B = (L, R, \omega, \lambda, \rho)$ be a bimachine defining some total function f. Then we have both $L \sqsubseteq L_0$ and $R \sqsubseteq R_0$.*

Proof: Let us show $R \sqsubseteq R_0$ since the proof is the same for L_0. Let $u, v \in \Sigma^*$ such that $u \sim_R v$. It means that $r_0 \xrightarrow{u} r$ and $r_0 \xrightarrow{v} r$ for some $r \in R$. Let $w = \sigma_1 \dots \sigma_n \in \Sigma^n$ and let $l_0 \xrightarrow{\sigma_1} l_1 \dots l_{n-1} \xrightarrow{\sigma_n} l_n$ be the execution of w on L. Similarly, let $r_n \xrightarrow{\sigma_1} r_{n-1} \dots r_1 \xrightarrow{\sigma_n} r$ be the execution of w on R from r. Then

$$\begin{aligned} \|f(wu), f(wv)\| &= \|\lambda(r_n)\omega(l_0, w, r)\omega(l_n, u, r_0)\rho(l), \lambda(r_n)\omega(l_0, w, r)\omega(l_n, v, r_0)\rho(l')\| \\ &= \|\omega(l_n, u, r_0)\rho(l), \omega(l_n, v, r_0)\rho(l')\| \\ &\leq k(|u| + |v| + 1) \end{aligned}$$

with k being the maximum size of an output in B. Hence $u \sim_{R_0} v$. □

In the following Proposition, we show that L^0 is maximal amongst the set of minimal left automata, but before we need the Lemma:

Lemma 5.3.1. *Let B be a bimachine with L as its left automaton and R as its right automaton. Let L' be finer than L and R' be finer than R. Then there is a bimachine B' with L' and R' as its automata and equivalent to B.*

Proof: Since L' and R' are finer than L and R, respectively, the bimachine B' can behave as B by disregarding the finer information on its states. □

Proposition 5.3.2. *Let f be a total rational function. Let R be a right automaton finer than R_0. Let L^R be the left automaton of the bimachine B^R. Then*

$$L^0 \sqsubseteq L^R$$

Proof: This follows from Theorem 5.3.1: Since $R \sqsubseteq R_0$, there is, according to Lemma 5.3.1, a bimachine defining f with L^0 and R as its automata and we have $L^0 \sqsubseteq L^R$. □

For a complete function f we have proven that L^0 is maximal amongst the set of minimal left automata. This means that if we want to show that f is definable by some bimachine with a $\mathbf{V}$-automaton as its left automaton, we only have to check if the finite set of automaton coarser than L^0 (and finer than L_0) contains a $\mathbf{V}$-automaton.

If there exists a $\mathbf{V}$-automaton coarser than L^0 and finer than L_0, then we define $L_{\mathbf{V}}$ as the finest left $\mathbf{V}$-automaton such that $L^0 \sqsubseteq L_{\mathbf{V}} \sqsubseteq L_0$. Finally we define $R^{\mathbf{V}} = R^{L_{\mathbf{V}}}$ as the right automaton of the bimachine $B^{L_{\mathbf{V}}}$.

Before moving on, let us show the following lemma:

Lemma 5.3.2. *Let $\sim_1 \sqsubseteq \sim_2$ be two right (resp. left) congruences. Let $\equiv_1$, $\equiv_2$ denote the transition congruences of the left (resp. right) automata associated with $\sim_1$ and $\sim_2$, respectively. Then $\equiv_1$ is finer than $\equiv_2$.*

Proof: Let us assume that $\sim_1$, $\sim_2$ are left congruences, the proof being completely symmetrical. We have that $u \equiv_i v$ if and only if for all $w \in \Sigma^*$, $wu \sim_i wv$. Hence we can easily see that $u \equiv_1 v \Rightarrow u \equiv_2 v$. □

We can now claim the desired property:

Proposition 5.3.3. *Let $\boldsymbol{V}$ be a monoid variety. Let f be a total rational function. The function f is definable by an unambiguous $\boldsymbol{V}$-NFT if and only if $R^{\mathbf{V}}$ is defined and is a $\boldsymbol{V}$-automaton.*

Proof: Let f be a total function, let us assume that there exists a $\mathbf{V}$-NFT defining f. According to Proposition 5.1.1 there exist a $\mathbf{V}$-bimachine $B = (L, R, \omega, \lambda, \rho)$ defining f. Let L^R be the left automaton of the bimachine B^R with R as its right automaton. Then, according to Theorem 5.3.1, we have $L \sqsubseteq L^R$, hence from Lemma 5.3.2, L^R is a $\mathbf{V}$-automaton. We have $L^0 \sqsubseteq L_{\mathbf{V}} \sqsubseteq L^R \sqsubseteq L_0$. Let $R' = R^{(L^R)}$ be the right automaton of the bimachine $B^{(L^R)}$ with L^R as its left automaton. We have, again according to Theorem 5.3.1, $R \sqsubseteq R'$, hence R' is also a $\mathbf{V}$-automaton. We have $R^0 \sqsubseteq R' \sqsubseteq R^{\mathbf{V}} \sqsubseteq R_0$. Hence according to Lemma 5.3.2, $R^{\mathbf{V}}$ is also a $\mathbf{V}$-automaton.

Conversely, we have in particular $B^{L_{\mathbf{V}}}$ a $\mathbf{V}$-bimachine defining the total function f hence, according to Proposition 5.1.1, we can obtain a $\mathbf{V}$-NFT defining f. □

Remark 5.3.1. In particular, total functions are completely characterized by a finite set of canonical bimachines in the following sense: Let f be a complete function defined by a bimachine B with L and R as its left and right automata. Then there exists a bimachine B_{min} defining f with L_{min} and R_{min} as its left and right automata with $L \sqsubseteq L_{min}$ and $R \sqsubseteq R_{min}$ such that: $L^0 \sqsubseteq L_{min} \sqsubseteq L_0$ and $R^0 \sqsubseteq R_{min} \sqsubseteq R_0$.

5.3.2 Partial functions

Now that we have the result for total function, we need to extend it to partial function. Let $f : \Sigma^* \to \Sigma^*$ be a (partial) rational function. We define the completion of f:

$$\begin{array}{rccl} \bar{f}: & \Sigma^* & \to & (\Sigma \uplus \{\bot\})^* \\ & u & \mapsto & \begin{cases} f(u) & \text{if } u \in \mathrm{dom}(f) \\ \bot & \text{otherwise} \end{cases} \end{array}$$

Lemma 5.3.3. *Let $\mathbf{V}$ be a monoid variety. Let f be a rational function such that* $\mathrm{dom}(f)$ *is a $\mathbf{V}$-language. The function f is definable by an unambiguous $\mathbf{V}$-NFT if and only if $\bar{f}$ is definable by an unambiguous $\mathbf{V}$-NFT.*

Proof: Let us assume f is definable by an unambiguous $\mathbf{V}$-NFT T. Then there is a $\mathbf{V}$-automaton recognizing the complement of $\mathrm{dom}(f)$. We can easily see that the complement of a $\mathbf{V}$-language is a $\mathbf{V}$-language: If M is a monoid in $\mathbf{V}$ such that $\mathrm{dom}(f) = \mu^{-1}(P)$ for some morphism μ and $P \subseteq M$, then $\Sigma^* \backslash \mathrm{dom}(f) = \mu^{-1}(M \backslash P)$. By adding to T a deterministic transducer that maps any word not in the domain of f to $\bot$, we obtain an unambiguous $\mathbf{V}$-NFT defining $\bar{f}$.

Conversely, let us assume $\bar{f}$ is definable by a $\mathbf{V}$-NFT. Since $\bar{f}$ is total, according to Proposition 5.1.1 f is definable by a $\mathbf{V}$-bimachine. Then by restricting the output function to values not equal to $\bot$, we obtain a $\mathbf{V}$-bimachine defining f. According to Proposition 5.1.1, again, f is definable by a $\mathbf{V}$-NFT. □

Remark 5.3.2. We have reduced the problem of deciding whether a function is definable by an unambiguous $\mathbf{V}$-NFT to the membership problem of $\mathbf{V}$.

Theorem 5.3.2. *Let $\mathbf{V}$ be a monoid variety. Let us assume that the membership problem is decidable for $\mathbf{V}$. Then there is an algorithm to decide if a function defined by an NFT is definable by an unambiguous $\mathbf{V}$-NFT.*

Proof. According to Proposition 5.1.1 and Lemma 5.3.3, f is definable by an unambiguous $\mathbf{V}$-NFT iff $\bar{f}$ is definable by a $\mathbf{V}$-NFT and the domain of f is a $\mathbf{V}$-language iff $\bar{f}$ is definable by a $\mathbf{V}$-bimachine and the domain of f is a $\mathbf{V}$-language.

The construction goes as follows: First we check that the domain of f is a $\mathbf{V}$-language. We construct a NFT defining $\bar{f}$. Then we construct the bimachine $B^{L_{\mathbf{V}}}$, if it exists, and check if $R^{\mathbf{V}}$ is a $\mathbf{V}$-automaton. If it is the case we use the construction of Lemma 5.3.3 to obtain a $\mathbf{V}$-bimachine defining f. Finally, from the construction of Proposition 5.1.1 we obtain an unambiguous $\mathbf{V}$-NFT defining f

□

Chapter 6

Remarks, examples and counter-examples

6.1 Monoid varieties

The monoid varieties we consider most often are given in Figure 6.1, which is taken from [Dar15]. We also give the corresponding logical and language characterizations. The equalities are given with the symbol ω which can be seen as the *idempotent power* of a finite monoid M, *i.e.* the smallest integer n such that any element to the n is idempotent. These equalities, called *profinite equalities*, also characterize the monoid varieties. For a proper definition we refer the reader to [Pin97].

Remark 6.1.1. Since one can determine the idempotent power of a finite monoid, we can see that the membership problem is decidable for varieties defined by a finite set of profinite equalities. This is true for the varieties listed in Figure 6.1 and in particular for the variety of aperiodic monoids.

6.2 Determinization

Here we see that, in general, determinization of transducers does not preserve the equalities of the transition monoid.

Example 6.2.1. Let us consider $\mathbf{I}$ the variety of idempotent monoids, *i.e.* satisfying $x = x^2$. Let $\Sigma = \{a, b\}$. We define the function:

$$f : a^n \mapsto \begin{cases} a \text{ if } n = 1 \\ b \text{ if } n > 1 \end{cases}$$

Variety	Equations	Logic	Languages
Finite Monoids	-	MSO	Rational
Commutative **Com**	$xy = yx$	MSO[=]	Commutative
Aperiodic **A**	$x^\omega = x^{\omega+1}$	FO	Star-free
$\mathcal{D}$-Aperiodic **DA**	$(xyz)^\omega y (xyz)^\omega = (xyz)^\omega$	$\mathrm{FO}^2 = \Delta_2$	Unambiguous
$\mathbf{J}_1$	$x = x^2, xy = yx$	FO^1	-
$\mathcal{J}$-trivial **J**	$y(xy)^\omega = (xy)^\omega = (xy)^\omega x$	$\mathcal{B}\Sigma_1$	Piecewise testable

Figure 6.1: Monoid varieties

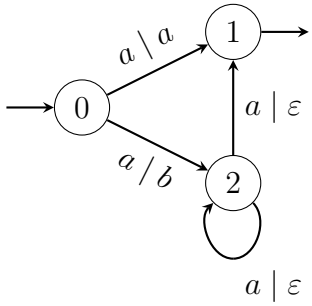


Figure 6.2: **I**-NFT.

The NFT of Figure 6.2 defines f and verifies $x = x^2$.

Let us assume that T' is an **I**-DFT defining f. Then we can decompose $f(a) = ut$ and $f(a^2) = uvt$, with $q_0 \xrightarrow{a|u} q \xrightarrow{a|v} q$ and t is the terminal output corresponding to q. Then $u = t = f(a) = \varepsilon$, which yields the contradiction.

Example 6.2.2. There is a sequential function definable by a **Com**-NFT that is not definable by any **Com**-DFT. Let $\Sigma = \{a, b, c\}$. We define the function f:

$$\begin{array}{rcl} ab & \mapsto & a \\ aab & \mapsto & a \\ aba & \mapsto & ab \\ ba & \mapsto & c \\ baa & \mapsto & c \end{array}$$

The NFT of Figure 6.3 defines f and verifies $xy = yx$.

Let us assume that T' is a DFT defining f and verifying $xy = yx$. Then we can decompose $f(ab) = u_1 t_1$,$f(ba) = u_2 t_1$, $f(aba) = u_1 v t_2$ and $f(baa) = u_2 v t_2$ with

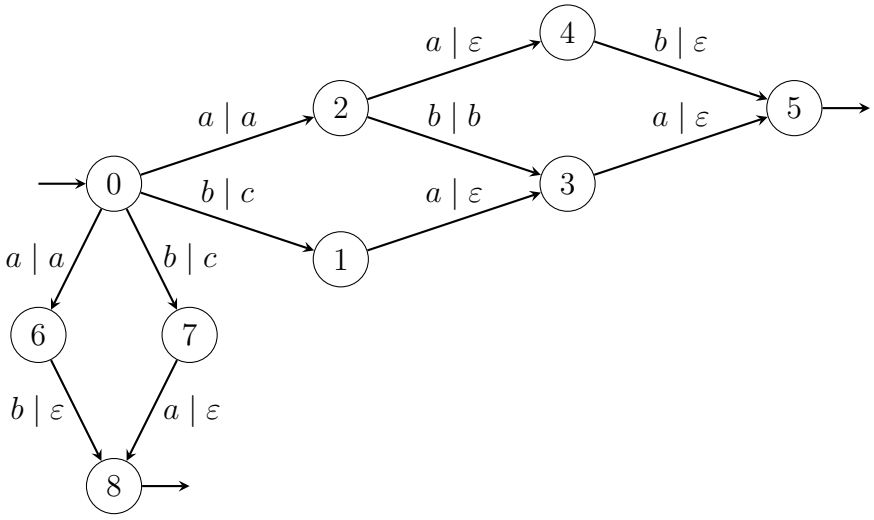


Figure 6.3: **Com**-NFT.

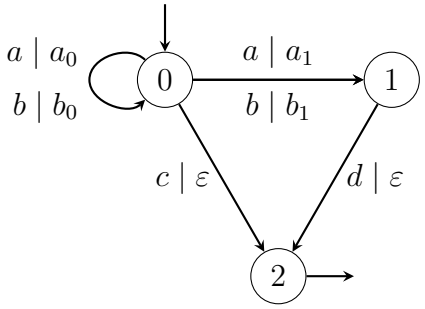


Figure 6.4: **J**-NFT.

$q_0 \xrightarrow{ab|u_1} q_1 \xrightarrow{a|v} q_2$, $q_0 \xrightarrow{ba|u_2} q_1$ and t_1, t_2 are the terminal outputs corresponding to q_1 and q_2, respectively. Since $f(aba)$ and $f(baa)$ have no letter in common, we have $v = t_2 = \varepsilon$. Hence $u_1 = ab$, and we have a contradiction.

Example 6.2.3. There is a sequential function definable by a **J**-NFT that is not definable by any **J**-DFT. Let $\Sigma = \{a, a_0, a_1, b, b_0, b_1, c, d\}$. The **J**-NFT of Figure 6.4 defines a subsequential function f.

Let us assume that T' is a **J**-DFT defining f. Then we can decompose $f((ab)^\omega c) = ut_c$, $f((ab)^\omega ac) = u\alpha t_c$ and $f((ab)^\omega ad) = u\alpha t_d$ with $q_0 \xrightarrow{(ab)^\omega|u} q \xrightarrow{a|\alpha} q$. Since $f((ab)^\omega c)$ and $f((ab)^\omega ac)$ have no common suffix, we have $t_c = \epsilon$. Hence we obtain that $f((ab)^\omega ac)$ is a prefix of $f((ab)^\omega ad)$ which gives the contradiction.

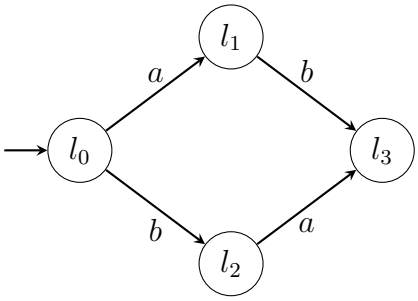


Figure 6.5: Left automaton of the bimachine B.

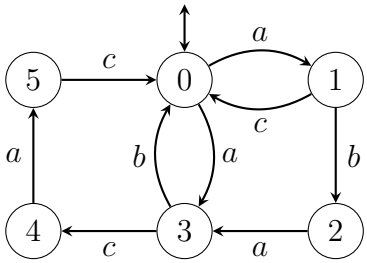


Figure 6.6: Aperiodic NFA.

6.3 V-domain

Let $\Sigma = \{a, b\}$. We give an example of a **Com**-bimachine $B = (L, R, \omega, \lambda, \rho)$ such that its domain is not a commutative language. The right automaton is the complete automaton with one state. The left automaton is given in Figure 6.5. The functions λ and ρ are constant and equal to ε. We define $\omega(l_0, a, r_0) = \omega(l_1, b, r_0) = \varepsilon$ and ω is not defined otherwise. Then the domain of the function is equal to $\{ab\}$ which is not a commutative language.

6.4 Disambiguation

Example 6.4.1. It has been shown that any rational function is definable by some unambiguous transducer. Moreover there is for transducers a disambiguation algorithm which yields an unambiguous transducer with an exponential number of states. Let $\Sigma = \{a, b\}$. We show in Figure 6.6 an example of an aperiodic automaton for which the algorithm described in [FGR12] never yields an aperiodic automaton.

As you can see this automaton is aperiodic, yet not counter-free. For state 0 there are two possible transitions reading letter a. If we choose $(0, a, 1) <_\delta (0, a, 3)$

then in the disambiguated automaton we have

$$(0, \varnothing) \xrightarrow{(ab)^k} (2, \varnothing)$$

If and only if k is odd. Similarly if $(0, a, 3) <_\delta (0, a, 1)$ then

$$(0, \varnothing) \xrightarrow{(ac)^k} (4, \varnothing)$$

If and only if k is odd. In both cases the obtained automaton is periodic.

Conclusion

Using techniques inspired by [Cho03, RS91], we have been able to extend decidability results for varieties of rational languages to varieties of rational functions. In the deterministic case the notion of minimal DFT gives a natural way to decide if a subsequential functions belongs to a given variety. In the non-deterministic case we need to consider not one minimal machine, but a finite set of canonical bimachines and if one of these machines satisfies the desired algebraic property, then the function it defines belongs to the corresponding variety.

Some of the questions that remain are for instance the complexity of the decision procedures described in this report. The question of the equivalence between $\mathbf{V}$-NFT and unambiguous $\mathbf{V}$-NFT for a general variety also remains unanswered.

We foresee several interesting directions in which the techniques of this report can be taken. The first one would be to try to extend the results to the more general class of regular functions, defined by two-way transducers [EH01] and streaming string transducers [AČ10], or some intermediate class. A second possible direction is the generalization to relations, with probably some condition of finite value, for instance by bounding the ambiguity of the transducers. Another extension would be to consider infinite words and see for instance if some minimization is possible for DFTs on infinite words. There is also the domain of weighted automata for which common techniques are shared with transducers [FGR14].

Bibliography

[AČ10] Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, pages 1–12, 2010.

[BB79] Jean Berstel and Luc Boasson. Transductions and context-free languages. *Ed. Teubner*, pages 1–278, 1979.

[BC02] Marie-Pierre Béal and Olivier Carton. Determinization of transducers over finite and infinite words. *Theor. Comput. Sci.*, 289(1):225–251, 2002.

[Büc60] Julius Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[CD15] Olivier Carton and Luc Dartois. Aperiodic two-way transducers and FO-transductions. *CSL*, 2015.

[Cho03] Christian Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003.

[Dar15] Luc Dartois. *Méthodes algébriques pour la théorie des automates*. PhD thesis, Université Paris-Diderot, École doctorale Paris Centre, 2015.

[DG08] Volker Diekert and Paul Gastin. First-order definable languages. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*., pages 261–306, 2008.

[EF95] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite model theory*. Perspectives in Mathematical Logic. Springer, 1995.

[EH01] Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.

[Elg61] Calvin Creston Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):pp. 21–51, 1961.

[ES76] Samuel Eilenberg and Marcel-Paul Schützenberger. On pseudovarieties. *Advances in Mathematics*, 19(3):413 – 418, 1976.

[FGR12] Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Quantitative languages defined by functional automata. In *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, pages 132–146, 2012.

[FGR14] Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Finite-valued weighted automata. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 133–145, 2014.

[Fil15] Emmanuel Filiot. Logic-automata connections for transformations. In *Logic and Its Applications - 6th Indian Conference, ICLA 2015, Mumbai, India, January 8-10, 2015. Proceedings*, pages 30–57, 2015.

[FKT14] Emmanuel Filiot, Shankara Narayanan Krishna, and Ashutosh Trivedi. First-order definable string transformations. *CoRR*, abs/1406.7824, 2014.

[MP71] Robert McNaughton and Seymour Papert. *Counter-free automata*. M.I.T. Press Cambridge, Mass, 1971.

[MSTV06] Pierre McKenzie, Thomas Schwentick, Denis Thérien, and Heribert Vollmer. The many faces of a translation. *J. Comput. Syst. Sci.*, 72(1):163–179, 2006.

[Ner58] A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):pp. 541–544, 1958.

[Pin97] Jean-Éric Pin. Syntactic semigroups. In *Word, Language, Grammar*, volume 1 of *Handbook of Formal Languages*, chapter 10, pages 679–738. Springer, 1997.

[RS91] Christophe Reutenauer and Marcel-Paul Schützenberger. Minimization of rational word functions. *SIAM J. Comput.*, 20(4):669–685, 1991.

[RS95] Christophe Reutenauer and Marcel-Paul Schützenberger. Variétés et fonctions rationnelles. *Theoretical Computer Science*, 145(12):229 – 240, 1995.

[Sak09] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[Sch61] Marcel-Paul Schützenberger. A remark on finite transducers. *Information and Control*, 4(2-3):185–196, 1961.

[Sch65] Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

[Str96] Howard Straubing. Finite models, automata, and circuit complexity. In *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop, January 14-17, 1996, Princeton University*, pages 63–96, 1996.

[Tra61] Boris Avraamovich Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961. In Russian.